Australian Government
Department of Defence
Defence Science and
Technology Organisation

Automatic extraction of 3D models from an airborne video sequence

Tristrom Cooke

Intelligence, Surveillance and Reconnaissance Division

Defence Science and Technology Organisation

DSTO–TR–2095

ABSTRACT

One method for accurately georegistering a video sequence from an airborne platform is to transform the video to the same coordinate system as some reference imagery that is already georeferenced. This transformation will be dependent upon the 3D structure within the scene, which is not known a priori. The current report examines several aspects of the construction of a 3D model from a video sequence, which may then be used for registration. The topics examined include: extraction of useful features (points, lines, or planes) from the images, determination of a sparse 3D model and camera motion model in cases where data may be missing, a method for estimating the depth at every pixel within a video frame, and finally an analysis of the errors at each step of the model construction process.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JAN 2008		**00-00-2008 to 00-00-2008**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Automatic Extraction of 3D Models From an Airborne Video Sequence	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Australia Government,Department of Defense,Technology Organisation,Australia,**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**65**	

Published by

Defence Science and Technology Organisation
PO Box 1500
Edinburgh, South Australia 5111, Australia

Telephone: (08) 8259 5555
Facsimile: (08) 8259 6567

AR No. 014-090
January 2008

APPROVED FOR PUBLIC RELEASE

Automatic extraction of 3D models from an airborne video sequence

EXECUTIVE SUMMARY

Information from airborne video can only be exploited when the scene imaged by the sensor can be associated with a point on the ground. While the video stream may contain embedded metadata, providing information such as platform position and camera orientation, this is sufficient to give only a crude geolocation for the imagery. More precise positioning can be obtained by matching the video imagery against an image (such as from an aerial survey) which has already been accurately georeferenced. Automatic matching is hampered due to the fact that the video and the image were likely taken from different camera positions.

Most existing image registration algorithms assume that the two images to be registered differ by a simple linear, quadratic, or perspective transformation. Images taken from different perspectives will also have a dependence on the depth of the scene, which is not easily parameterised. One method for dealing with this is to automatically construct a 3D model from the video data, and then reproject it as it would be seen from point of view of the reference image. This would leave only a simple translation between the images. The current report describes new and existing methods which may be used for the automatic extraction of a 3D model from airborne video imagery. An overview of the system, which uses such a model for the georegistration of video imagery, is the subject of a separate report [8].

There are three key steps to the construction of a 3D model, as described in this report. The first is the detection and tracking of features through the video sequence. Corner detectors have mostly been described previously in DSTO-TR-1759, so this report has some additional information on a few corner detectors not previously tested, and a short section on line detection.

The second step in constructing a 3D model is to estimate the camera pose for each frame and the 3D positions of each of the tracked features. When all points are successfully tracked over all frames, the factorisation method can be used to estimate all of the required parameters. This report considers several methods for dealing with the more realistic case where large amounts of data are missing. The most successful of these seems to use a robust modification of Tomasi and Kanade's hallucination algorithm, followed by iterations of Shum's method. A factorisation method based on tracked lines, instead of tracked points, is also mentioned.

The output of the second step is a sparse 3D model. To successfully model the entire scene, the depth if the scene at each image pixel is required. This report considers several approaches including segmenting the corner points into facets, and stereo dense matching algorithms. Dense matching using graph-cuts was by far the most successful of the tested techniques and produces a model which is expected to be useful for any subsequent 2D image registration step.

Following the descriptions of how to obtain a 3D model from a video sequence, this report also describes a framework for quantifying the errors in the construction of the

model. Simulations of this framework using real airborne video imagery indicate that the resulting error estimates will be similar to the actual errors.

Author

Tristrom Cooke
Information, Surveillance and Reconnaisance Division

This author obtained a B.Eng (Hons) in Electrical Engineering from the University of South Australia in 1992, a B.Sc (Hons) in Applied Mathematics from the University of Adelaide in 1995, and completed a PhD in Applied Mathematics (also at the University of Adelaide) in the area of thermo-elasticity at the end of 1998. He was then employed by CSSIP (CRC for Sensor Signals and Information Processing) until 2005, where he has mostly worked on projects relating to recognition of targets in both synthetic aperture and inverse synethic aperture radar imagery. He is now full time employee in ISRD of DSTO, where he is working on structure from motion.

Contents

Figures

1 Introduction

There are a number of difficulties involved in fusing the information from different sources of imagery. One of the major difficulties is in determining which pixel in one image corresponds to that in the other, or the registration problem. There are a number of confounding factors such as variations in sensor type and environments, which are mostly cosmetic factors which do not structurally affect the images. A more difficult problem to overcome is due to the structure of the scene appearing different due to a difference in viewpoint of the two platforms. An overview of a system for registering imagery taken from airborne video against some reference imagery is described by [8]. One of the key steps of this system is to recover a 3D model of a scene from the video sequence taken using a moving camera, the structure from motion problem. This report provides more thorough technical information on some of the steps required to accomplish this, along with details of alternative methods that have also been considered as a part of this process.

Structure from motion problems have been studied for some time, and the established methods of solution usually consist of four major steps. The first step is to extract useful features (points, lines, or planes) from each of the images. Then, each of the features is tracked throughout the image sequence. Following this, a 3D model and camera motion model are determined which are consistent with the measured feature positions. Finally, high level information about the scene (such as the general structure of buildings usually consisting of plane facets intersecting perpendicularly) is used to refine the model.

Section 2 deals with the first step of the structure from motion process, which is the detection of features. Detection and tracking corner points were addressed in a previous report [7], and so have not been covered in depth here. This section gives a quick overview of the phase congruency detector which was not tested in the previous report. It then goes on to describe the detection of linear features in the image, and methods for jointly detecting lines and the points corresponding to the intersection of lines.

From a set of detected and tracked features, it is then required to find the combination of structure and motion which would be consistent with these measurements. Section 3 describes a number of methods for determining the structure and motion using various camera models (such as orthographic or perspective) from measurements taken of both points and lines. Methods for dealing with obscuration and loss of track of features have also been discussed.

The output of the previous step is a collection of geometric primitives (mostly points and lines) scattered throughout 3D space. The final step involves interpretation of these points under the assumption that the object being imaged is a building, or some other man-made structure. This involves the arrangement of the points and lines into higher order structures such as planes. This data can then be used to extract textures from the image and reconstruct views of the scene from camera angles not available in the original video sequence. All of these topics have been discussed in Section 4 on post-processing.

The final step, 2D registration, is not examined in any detail in this report. It is touched upon in Section 6, however, which describes a framework in which errors in the entire video registration process may be automatically estimated. Some conclusions and recommendations for future work are then summarised in Section 7.

2 Feature detection

The first step in the processing of the video data is to register between consecutive frames in the video sequence. The most common way for accomplishing this is feature detection, where a set of prominent objects are detected in one frame, and matched to similar looking objects in the next frame. This correspondence will eventually allow 3D information about the detected objects and camera positions to be extracted.

The types of features detected may also influence the amount of understanding of the scene. For instance, corner points and edges can indicate the positions of the joins between two planar facets of a scene, and the identification of planar regions can give the approximate topology of a scene without even any additional information from other video frames. Accurate detection of any of these features could theoretically allow wire-frame models of the scene to be constructed without the necessity for dense matching methods. Some methods for attempting this are discussed in Section 4. The combination of the detection methods, and the subsequent post-processing stages have, so far, been of limited success.

This section describes three types of feature detection. The first is the detection of corners using the phase congruency detector. A more comprehensive review of other corner detection methods is described in another report [7]. The second subsection describes the detection of linear features, and the last subsection describes a method which jointly detects corners and lines.

2.1 Phase congruency corner detection

The phase congruency detector, described by Kovesi [18], is related to the local energy detector presented in a previous report [7]. The local one dimensional energy of a point in an image is given by

$$F(x, y, \varphi_0, \sigma) = \sqrt{(G_{even}(\varphi_0, \sigma) * f)^2(x, y) + (G_{odd}(\varphi_0, \sigma) * f)^2(x, y)}$$

where G_{even} and G_{odd} are filters for detecting intensity peaks and edges respectively, φ_0 is an edge orientation and σ is a scale parameter. This local energy emphasises step edges, as well as lines in the image. Venkatesh and Owens [37], have apparently shown that the local energy can be expressed as

$$F = P(x, y) \sum_n A_n$$

where the A_n are the amplitudes of the image frequency spectrum and P is the phase congruency, which is a measure of the consistency of the phase of the Fourier components of the image at a given point, as given by the equation

$$P(x, y) = \max_{\phi_0} \left(\frac{\sum_n A_n \cos(\phi_n(x, y) - \phi_0)}{\sum_n A_n} \right).$$

where ϕ_n is the phase of the nth frequency at the point (x, y). The phase congruency is a number between zero and one, and will be a measure of the edginess at a point in the image, which is invariant to intensity scaling. This invariance however also means that it is very sensitive to image noise. It also produces an artificially high estimate for the phase congruency when there is only one frequency present. To get around these difficulties, Kovesi suggested a modified phase congruency given by

$$P_2(x, y) = W(x, y) \max_{\phi_0} \left(\frac{\sum_n \max(A_n \cos(\phi_n(x, y) - \phi_0) - T, 0)}{\sum_n A_n} \right).$$

which includes a factor $W(x, y)$ measuring the spread of frequency components, and ignores those image components below a threshold T corresponding to the image noise, which can be automatically estimated from the image.

Like the local energy, the phase congruency can be calculated for a number of orientations. The variation of the phase congruency with angle provides information about the type and orientation of the feature. In Kovesi's follow-up paper [19], this variation is described by second order moments

$$\begin{aligned} M_{xx} &= \sum_{\varphi} (P_2(x, y, \varphi) \cos(\varphi))^2, \\ M_{xy} &= \sum_{\varphi} 2P_2^2(x, y, \varphi) \cos(\varphi) \sin(\varphi), \\ M_{yy} &= \sum_{\varphi} (P_2(x, y, \varphi) \sin(\varphi))^2. \end{aligned}$$

From these, the two principal moments and their orientations can be determined. A corner point may be expected to be edge-like (i.e. have high phase congruency) in more than one direction, so the smallest of the principal moments may be used as a measure of the cornerness of a particular point within the image. Similarly, the larger principal moment would measure how edge-like the point was.

To test the performance of this corner detector, it was applied to 51 frames from an infra-red video sequence of an aircraft control tower at Parafield, in South Australia, taken from an airborne MX20 infrared sensor. A set of ground-truth corners were marked for each of the 51 frames of size 700×480, and ROC curves were constructed to measure the detection characteristics, as described in a previous report [7]. Figure 1 compares the performance of the phase congruency corner detector with five of the best performing detectors from previous tests. For the tests, Kovesi's own MATLAB code was used with the default parameters. The results show that this method is competitive with the other good detectors, although it is not consistently better (or worse) than any of the other detectors in this graph.

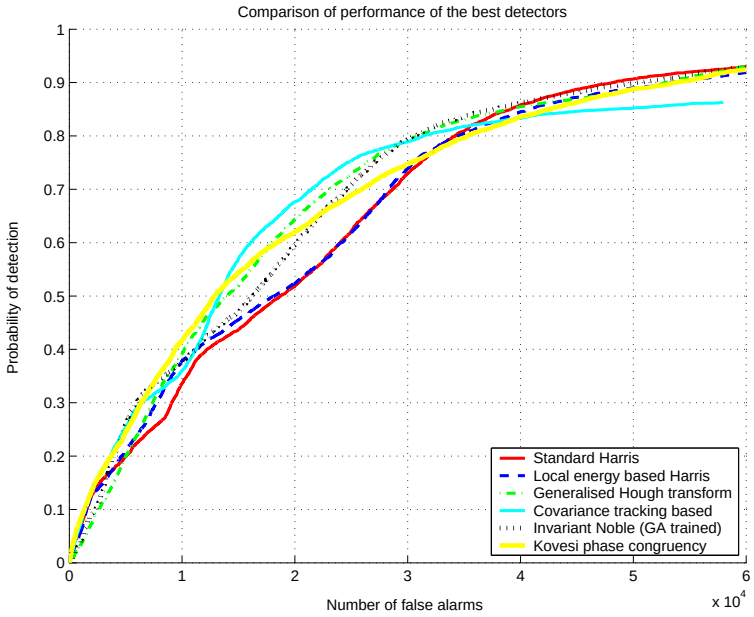

Figure 1: *Comparison of performance of corner detectors*

2.2 The FAST corner detector

Another set of corner detectors evaluated here are based on the FAST detector [24], which has received attention due to its combination of speed and performance. The FAST detector is listed as a "key corner detector" in a recent comprehensive review of local features [20].

FAST classifies a central pixel based on a ring (3 pixels radius, 1 pixel wide) of pixels surrounding it. If there is a consecutive set of N pixels, either larger than or less than the central pixel intensity by some threshold T, then the central pixel is labelled as a corner. The original FAST detector [23] used $N = 12$ and relied on a small manually chosen set of comparisons to classify corners very quickly, while other values of N relied on a slower test. In a later paper [24], this was generalised to other values of N by using entropy based methods to construct a decision tree, followed by evaluating the code on a training image to prune comparisons that were not needed. The resulting FAST-N detectors classified in a similar way to the original formulation, but required significantly fewer comparisons. Using the repeatability of detected corners as a measure of performance, it was found that FAST-9 was the best of these detectors.

A third, as yet unpublished, paper [25] extends these results further with the FASTER detector, where the ER stands for Enhanced Repeatability. This is a modification of the FAST-N detector, where the ring of outer pixels is thickened, and the particular set of comparisons is determined using simulated annealing to optimise a performance measure over some training images. The measure to be optimised is a combination of the number of points detected, the repeatability, and the complexity of the decision tree.

The standard implementation of FAST methods uses a fixed threshold T, and pro-

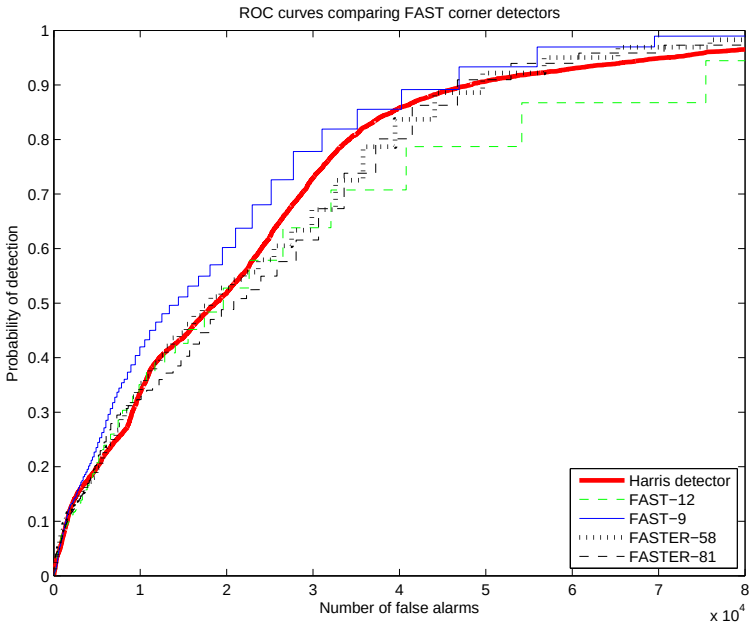

Figure 2: *Comparison of FAST corner detectors*

duces a binary classification at each point. This can be converted into a corner strength measurement by finding the maximum value of T which still results in a detection at that point, although this obviously reduces the algorithms' computational efficiency. Using this approach, the performance of the above corner detectors was evaluated on the Parafield airport data set, and the results are shown in Figure 2. The FAST-9 detector was found to be better than the original FAST in terms of repeatability, and this is also true in terms of detection performance for the Parafield data. In fact FAST-9, as implemented here without pruning, is the only detector, out of all those tested on this data, that has consistently performed better than the Harris detector for the entire ROC. For the FASTER algorithm, two different implementations (corresponding to different runs from the simulated annealing algorithm) were supplied by Edward Rosten. While these implementations were found to give better repeatability than FAST-9, the repeated points were not necessarily corner points, and for this data the detection performance was actually reduced.

2.3 Line detection

The previous subsection described several methods for the detection of corners within an image. These can be tracked between frames to provide 3D information on a set of points which are relatively useful in describing the scene. Detection of corner points, compared with other stable points, appears to be fairly difficult. This is largely due to most characterisations of a corner being on a local scale, where it is relatively likely for pixels to appear like corners by chance. To reduce this effect, it is necessary to consider more extended information using features such as lines. This subsection briefly compares several simple methods for detecting lines. The use of lines for extracting a 3D model of a scene is described in Subsection 3.4.

2.3.1 Hough transform detectors

The most frequently used methods for the detection of lines in images utilise the Hough (or Radon) transform. This transform represents a line in image space, parameterised by $(x, y) = (\rho\cos\theta + t\sin\theta, \rho\sin\theta - t\cos\theta)$ for varying t, as a single point (ρ, θ). The detection of bright or dark lines in an image is therefore equivalent to finding maxima or minima in Hough space. Frequently, however, linear structures in the image are not bright or dark lines, but edges. The detection of these edges is usually accomplished by edge enhancement, followed by a Hough transform to detect lines in the resulting edge image. This approach does not differentiate between edges that are parallel or perpendicular to a considered edge. A more discriminating approach therefore would be to apply a 1D edge filter in the ρ direction to the Hough transform of the original image. This detector, using the Radial Derivative of the Radon Transform (RDRT) has been successfully used to find faint trails in SAR imagery [6]. However, application of the same method to optical imagery failed due to false lines being generated from very short, bright and sharp edges. Two alternative approaches to RDRT for edge detection in optical imagery are now described.

Both of the methods suggested here are based on a new Hough measure. Firstly, binary edge images E_x and E_y are calculated in the x and y directions, based on an appropriately chosen edge threshold T. The threshold could be automatically selected for each image to keep some small fixed percentage of pixels. A combined Hough measure is calculated

$$H_{total}(\rho, \theta) = \cos^2(\theta)H_x(\rho, \theta) + \sin^2(\theta)H_y(\rho, \theta),$$

where H_x and H_y are the Hough transforms of the individual edge images. The resulting measure is similar to the RDRT in that it only considers the edge strength perpendicular to the considered line segment. Due to the binarisation of the edge images, it is largely independent of the actual edge strength. The resulting Hough transform is effectively a measure of the number of edge pixels lying on each line.

Method 1: For a notional line to be classified as part of an actual image line, it must be statistically different from randomly distributed pixels. The probability of any random point containing an edge point can be measured, and the number of points on the line will be binomially distributed (or approximately normal for sufficiently many points). Therefore a statistical hypothesis test may be applied to detect lines. The strength of the test chosen is another parameter of the test, but should be relatively independent of the imagery type.

Having detected one edge, it is not unlikely that others may also be present. These edges may be drowned out by the presence of the dominant line, so its effect is removed by zeroing the edge pixels corresponding to the detected line. The hypothesis testing is repeated until no more lines are detected.

In optical imagery, line segments of interest will usually not span the image, and will not be of an even size. Therefore, the detection is conducted on overlapping tiles of the image for a fixed sized window. This process is then repeated for a number of different scales. Such a multiscale approach had also been considered in the RDRT method, with

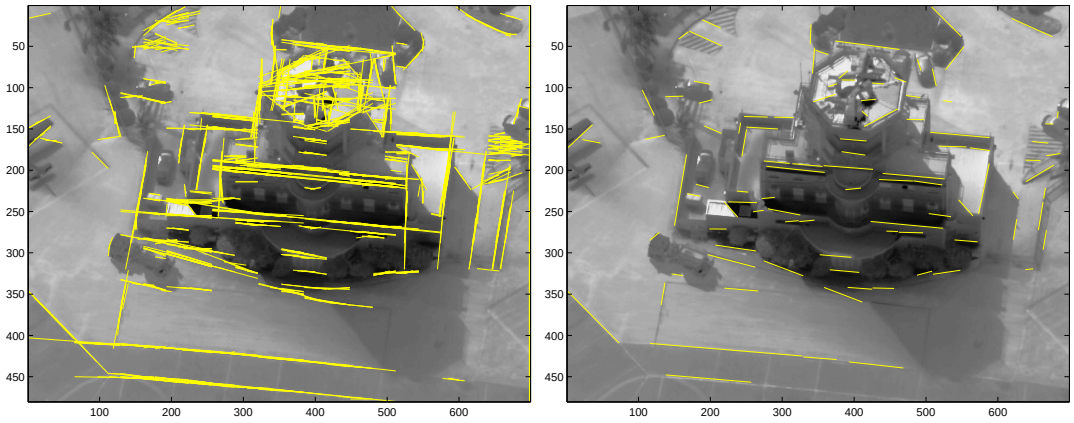

Figure 3: *Detection of lines using two Hough transform approaches*

some arbitrary factors used to modify the thresholds at the different scales. This is not a problem in the current method, as the same statistical significance of the test can be used at all scales.

Method 2: Instead of assessing the scene at multiple scales, it is possible to iteratively detect the strongest N lines present in the image directly from the Hough transform for the entire image. The segment of the line which corresponds to the edge of interest can then be determined by extracting all of the edge pixels in order along the current dominant line, and applying a median filter to fill in any gaps in the edge. The longest consecutive run of edge pixels localises the extent of the current detection. These edge pixels are then removed from the edge image, the Hough transform recalculated (for speed, this is only done for the small region about the previous detection), and then the process repeated until the quota of lines has been met.

Figure 3 shows the results from the multiscale Hough transform, and iterative detection approaches being applied to an image of Parafield airport control tower. In both cases, the edge threshold was set to 4, and for the iterated detection, the strongest 100 lines were detected. The first method results might have been improved by using the edge localisation technique from the second method to limit the detected lines, instead of assuming the line spanned the entire width of the window. This is especially true at the top of the control tower.

2.3.2 Burns' detector

An often used alternative to Hough transform methods for lines detection is the method of Burns, Hansen and Riseman [4]. This method first finds the set of pixels having an edge strength higher than some threshold. The orientation of the edge at each point is also determined. Neighbouring edge pixels that fall into the same orientation quantisation bin are likely to correspond to the same edge, so they are collected together to form connected

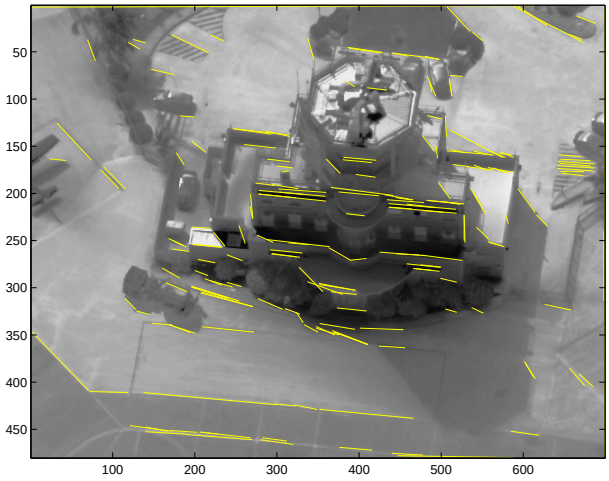

Figure 4: *Detection of lines using Burns' method*

edge curves. Edge curves below a certain size are discarded, and straight lines are fit to the remaining segments. The end-points of the lines are taken to correspond to the bounding-box of each of the segmented regions. Figure 4 shows the result of applying the Burns' line detector to the Parafield control tower image. The detected lines appear very similar to those produced by the iterated detection Hough method from the previous subsubsection. The effect of the parameters (minimum segment size and edge threshold) for the Burns method, however, seems to have a larger effect on the resulting detections, making tuning the parameters for a given image more difficult. The first Hough transform method only has one parameter to be chosen, and for this image appears relatively insensitive to that. Both Hough methods require significantly more computation than Burns' method.

2.4 Joint point/line detection

When only local information about a point is available, it is difficult to robustly detect corners because random variation within the imagery will also produce corner like structures. The corner points that are of most interest, however, will generally correspond to the intersection of building edges, at least two of which will be visible in the image. By detecting lines corresponding to edges concurrently with corner points, it is hoped that many of the fake corner points will be removed.

An initial implementation which attempts to jointly detect points and lines in an image is described below:

- **Get initial corner points:** The corner points are initialised using a standard detector.
- **Get initial line segments:** It is assumed that each corner belongs to two or more line segments, connected to other corners. Therefore, for every corner i, the line to its neighbour $j \in \mathcal{N}_i$ is examined. The edge strength between the corners is calculated perpendicular to the line, and the average fraction of edge pixels stronger

than some threshold is taken as the line strength. Each corner point, is assigned two line segments to neighbouring points, corresponding to the strongest of these lines.

- **Iteration:** The following is repeated until convergence.
  - **Update corner positions:** By examining the profile of each corner's connecting line segments, it is determined which direction will produce the largest increase in the strength of its two edges. The corner is then moved one pixel in this direction.
  - **Update neighbourhood:** All of the feature's neighbours are re-evaluated to determine if a stronger edge exists.
  - **Feature removal:** If a point has, mutual links to its neighbours that are in roughly opposite directions, and have no other line segments to them, then they are likely to be edge pixels rather than corner points. These are removed, with the two line segments being merged into a single edge. Similarly, if the segments from a point are in almost the same direction, one of them is removed, and a segment to the next strongest neighbour is added.
- **Display result:** The corner points are plotted, along with all associated line segments that are above a specified threshold.

Figure 5 shows the result of the above algorithm (with a fairly strict definition of collinearity) applied to the Parafield airport image. The first image shows the distribution of the strongest 794 corner pixels, as produced by the Shi-Tomasi detector. The distribution of features is initially quite haphazard, but after convergence most of the unimportant areas no longer contain features, and most of the principal lines have been captured to some extent. Although the initial positions of the features were corners by some measure, most of the detections have drifted towards lines, which are still probably more corner-like than some of the original detections.

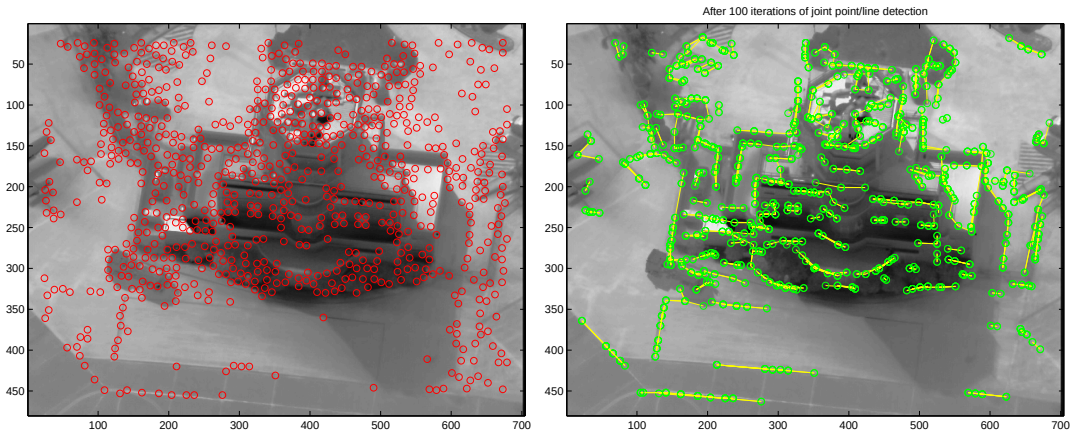

Figure 5: *Corner points and associated lines before, and after joint point/line detection*

3 Models for extracting structure and motion

If the same features of an object can be located in every image in a sequence, and a three-dimensional reconstruction is needed, there are many ways to approach the problem. A relatively simple group of algorithms for solving these problems is collectively referred to as factorisation methods. These rely on simplifications of the geometric model of the problem which allow a set of measurements, arranged into a measurement matrix, to be decomposed into structure and motion components using a singular value decomposition. Subsection 3.1 describes the rank 3 factorisation method of Tomasi and Kanade [31] for solving the structure from motion problem using a scaled orthographic camera model. This subsection sets up the notation and describes the camera models required for the following subsections. A more thorough examination of factorisation methods and their application to more complicated camera models can be found in another report [38].

Factorisation methods require that a set of features have been accurately tracked over the entire image frame. This assumption is usually false due to obscuration of features and tracker error. In order to still be able to use factorisation methods with this data, a set of methods were developed to fill in missing measurements by "hallucinating" or "imputing" them prior to factorisation. A number of such hallucination algorithms and other methods for dealing with missing data have been described in Subsection 3.2. This subsection also provides some numerical comparisons of these methods applied to a standard video data set. Some related methods are required to be used in cases where a feature is temporarily obscured, and the tracker picks it up again later as a new and separate track. In this example, the two tracks should be joined, and a way of automatically accomplishing this joining is described in Subsection 3.3. Finally, Subsection 3.4 briefly discusses a method for solving the structure from motion problem where linear features have been tracked, instead of points.

3.1 Factorisation

Given a set of features (either points, or lines) extracted from a video sequence, information about the relative position of the camera for each frame, as well as the 3D positions of the features, can be determined. The method used to estimate these parameters will depend on the camera model. Many are available but, to reduce sensitivity to measurement errors, it is suggested that the simplest model consistent with the data be used. For modelling structures from airborne video, the scene being imaged will generally be small compared to the distance from the camera, which means that perspective effects (such as parallel lines meeting at a vanishing point) will not be significant, and a scaled orthographic model can be used.

The scaled orthographic model was popularised by Kanade and Tomasi's rank 3 factorisation algorithm [31] for simultaneously obtaining the camera parameters and the 3D positions of the tracked points. This method is now described. The scaled orthographic model describes a set of n point measurements which are tracked over a set of m frames, and can be represented by a measurement matrix $\mathbf{M}$ given by

$$
\begin{aligned}
M &= \left[\begin{array}{cccc}
x_1(1) & x_2(1) & \dots & x_n(1) \\
x_1(2) & x_2(2) & \dots & x_n(2) \\
\dots & \dots & \dots & \dots \\
x_1(m) & x_2(m) & \dots & x_n(m) \\
\hline
y_1(1) & y_2(1) & \dots & y_n(1) \\
y_1(2) & y_2(2) & \dots & y_n(2) \\
\dots & \dots & \dots & \dots \\
y_1(m) & y_2(m) & \dots & y_n(m)
\end{array}\right] \\
&= \left[\begin{array}{cccc}
r_{xx}(1) & r_{xy}(1) & r_{xz}(1) & t_x(1) \\
r_{xx}(2) & r_{xy}(2) & r_{xz}(2) & t_x(2) \\
\dots & \dots & \dots & \dots \\
r_{xx}(m) & r_{xy}(m) & r_{xz}(m) & t_x(m) \\
\hline
r_{yx}(1) & r_{yy}(1) & r_{yz}(1) & t_y(1) \\
r_{yx}(2) & r_{yy}(2) & r_{yz}(2) & t_y(2) \\
\dots & \dots & \dots & \dots \\
r_{yx}(m) & r_{yy}(m) & r_{yz}(m) & t_y(m)
\end{array}\right]
\left[\begin{array}{cccc}
s_{x1} & s_{x2} & \dots & s_{xn} \\
s_{y1} & s_{y2} & \dots & s_{yn} \\
s_{z1} & s_{z2} & \dots & s_{zn} \\
1 & 1 & \dots & 1
\end{array}\right] = \mathbf{RS} + \mathbf{T} \qquad (1)
\end{aligned}
$$

where $(x_i(t), y_i(t))$ are the coordinates of the ith feature point in the tth image. The matrices $\mathbf{R}, \mathbf{S}, \mathbf{T}$ correspond to the camera, structure and translation matrices. When all of the entries within the measurement matrix are known, the translation matrix T (corresponding to the $t_{x/y}(t)$ elements within the expanded matrix) may be removed by subtracting the centre of mass of the features within each frame. The measurement matrix $\mathbf{M}$ will then have a rank of three. The scaled orthographic assumption implies that the camera matrix $\mathbf{R}$ have its rows $[r_{xx}(t), r_{xy}(t) \dots, r_{xz}(t)]$ and $[r_{yx}(t), r_{yy}(t) \dots, r_{yz}(t)]$ be orthogonal and of equal length for each frame t.

In practice, the measurement matrix will not be known exactly, due to measurement error. Instead, an estimate $\hat{\mathbf{M}}$ is available, from which the camera and structure matrices must be determined. The matrix can be written as a Singular Value Decomposition (SVD) $\hat{\mathbf{M}} = \mathbf{UDV}^T$ where U and V are orthogonal matrices and D is a diagonal matrix of singular values. Assuming the estimated matrix is close to the actual measurement matrix, then only the first three singular values will be of any significance. Zeroing out the remaining values to give $\mathbf{D}'$ gives the rank 3 matrix which is closest to the estimated matrix in a least squares sense. The camera and structure matrices will then correspond to $\mathbf{R} = \mathbf{U}\mathbf{D}'^{1/2}$ and $\mathbf{S} = \mathbf{D}'^{1/2}\mathbf{V}^T$.

The parameters $\mathbf{T}$ were found using the centre of mass of the corner points, and $\mathbf{R}$ and $\mathbf{S}$ which best satisfy equation (1) have been estimated using the SVD. These solutions, however, are not unique since

$$\mathbf{M} = \mathbf{RS} + \mathbf{T} = \mathbf{RA^{-1}}\,\mathbf{A}(\mathbf{S} + \mathbf{B}) + \mathbf{T} - \mathbf{RB}$$

for any invertible matrix $\mathbf{A}$ and translation matrix $\mathbf{B}$. In addition to this, the rows of the camera matrix $\mathbf{R}$ do not satisfy the orthogonality requirements. The matrix $\mathbf{A}$ can be chosen to satisfy this constraint, but there still remain ambiguities in the solution, including an arbitrary rotation, scaling, flipping and translation of the camera and 3D

structure. These ambiguities cannot be resolved without the availability of additional data.

3.2 Dealing with missing measurements

For a scaled orthographic camera model, the measured positions of the corner points in each frame will be related to their actual positions in 3D by the relation

$$M = RS + T$$

where R is a $2m \times 3$ sized camera matrix, S is a $3 \times n$ sized structure matrix, and T is a matrix of the translations for each image frame, each value filling a row. In the ideal case, probably the best method for determining the unknown parameters R,S and T is to use Kanade and Tomasi's rank 3 factorisation algorithm [31], as described in the previous subsection. In practical problems, however, many features are not detected in some views. This could be due to either imperfections in the feature detection and tracking or the feature becoming occluded. Only using measurements that have been tracked through all frames severely limits the amount of usable data, and can result in gross errors in the estimation of the model parameters. Several methods for estimating R, S and T when the measurement matrix M contains missing data are now considered.

3.2.1 Tomasi and Kanade's algorithm

A number of approaches have been proposed to handle the case where some elements of the matrix of observations **M** have missing values. Tomasi and Kanade's refer to their original approach for dealing with occlusions as a "hallucination" algorithm. This was described in an early 1992 Carnegie Mellon University report [33]. It assumed that a core sub-matrix , of size at least 6×3 was available for which all data was known, and factorisation was used to estimate camera parameters and structures for this subset of frames and points. If another frame contains more than three features in common with this submatrix, then

$$\begin{bmatrix} M_{sub} \\ m_{new-rows} \end{bmatrix} = R_{sub} \begin{bmatrix} S_{sub} \\ r_{new-frame} \end{bmatrix} + T_{ext},$$

where T_{ext} contains another of the repeated columns, and only $r_{new-frame}$ is unknown. This is now an over-determined linear set of equations for the new measurements in $m_{new-row}$, and so linear least squares may be used to estimate the 3D position of the new camera parameters, $r_{new-frame}$, and therefore the missing elements in $m_{new-rows}$ to give a new and larger filled sub-matrix. A similar process may be used to estimate the 3D coordinates of a new point appearing in at least two of the frames whose camera parameters are known. Therefore, the measurement sub-matrix may be grown by one row or column at a time until the matrix is filled.

The propagation of errors through the hallucination process will vary depending on the choice of initial sub-matrix, and the order in which rows and columns are added.

Tomasi and Kanade suggest that this should not make much difference although they do use a greedy method to choose the column or row with the greatest number of existing measurements at each update. They also suggest that the result should be used to initialise a steepest descent method for minimising the error between the measurements and values predicted from the model parameters.

3.2.2 Jacobs' method

Another frequently considered method for obtaining an initial estimate of the low-rank factorisation of a matrix is presented by Jacobs [16]. For the structure from motion case, he considers all pairs of image frames which have $k > 4$ tracked points in common. For each pair, a submatrix of filled values using these frames is then extracted. Since the original measurement matrix can be expressed as

$$M = RS + T = \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} S \\ 1 \end{bmatrix},$$

the $5 \times k$ matrix consisting of the filled sub-matrix, with an added row of ones, will remain rank 4. After an SVD, the last $k - 4$ columns of the resulting matrix will be the null-space n_i for the ith pair of image frames. Evidently, the full null-space for the complete measurement matrix has rank $n-4$, but due to the missing data, the only vectors guaranteed to lie in the null-space n_i are the ones which have zeros corresponding to the missing entries.

Once the null spaces n_i have been determined for each image pair, they are then averaged in a sense by combining them into a large matrix N. Since all of the null-spaces should be orthogonal to the principal values that are of interest, the principal values should lie in the null-space of the array of null-spaces. Therefore, by applying factorisation to N, and selecting the three least eigenvalues, the structure matrix S of the original matrix with missing data can be determined. The camera poses R and translation T may then be determined by a least squares fit to the original measurement data.

3.2.3 Shum's method

Another method that is frequently used for refining the factors of a matrix with missing data was given by Shum et al. [30]. The idea here is to iteratively solve for the camera matrix and translations assuming the structure is correct and then for the structure assuming that the camera poses are correct. These problems are linear, and only the measurements in a given frame are dependent on the parameters of the frame. This means that each iteration can be solved using a large number of small least squares problems to estimate the structure of each point and the camera parameters of each frame independently. The greatest difference between this method and iterative factorisation is that none of the hallucinated data is used in the estimation process. Instead, each of the measurements is assigned a unit weight if it is present, a zero weight otherwise, and then the least squares problem is solved for a weighted error. In theory, if it is known that some of the measurements were more accurate than others, different weights could be used for these points, but this is not generally used in practice.

3.2.4 Troyanskaya's method

Structure from motion is not the only application for which it is necessary to deal with missing measurement data. Troyanskaya et. al. [35] for instance considered missing values in a DNA micro-array. They considered two methods: a nearest neighbour and an SVD based imputation method. The nearest neighbour method worked by finding a measure of closeness between a row with missing data and other rows containing values in the same columns. The missing data was then imputed by taking a weighted average of results from the rows or columns for which the data is known. This type of imputation is not suitable for structure from motion, due to the structure of the missing data, amongst other reasons.

The second imputation method was an iterative update scheme. Here, a number of principal rows or columns of the initial matrix are extracted using an SVD. Troyanskaya et al. suggest about 15 for processing micro-arrays, but it makes sense that the dominant 4 vectors (as used by Chen and Suter [5]) or 3 plus a vector of ones be considered for structure from motion, where the rank of the true matrix is known. The known data can be expressed as a linear combination of the principal vectors which can be used for estimating the missing data.

When 4 vectors are used, the above approach is equivalent to applying factorisation to the initial estimate to obtain R, S, T and using this to update the missing entries in the measurement matrix. The process is continued until the solution converges. Chen and Suter [5] criticised this method because although it always converged, it would frequently not converge to the correct result. This has been confirmed using a MATLAB implementation of the algorithm with a set of simulations of tracked points from a cylinder, where the tracks were initialised and lost according to a Markov probability model. In fact, using this data, a number of the cases converged to wildly incorrect predictions for the missing data even though the initial predictions were chosen to be very close to the true values. Increasing the number of singular values used does not appear to add any stability to the procedure.

3.2.5 Dealing with outliers

While the previous methods have attempted to make estimation of parameters from the measurement matrix robust to noise, Huynh et al. [15] have tackled this from a different angle by reducing the weighting assigned to noisy measurements. This is done by updating the measurement matrix so that points further than expected from the current model are moved towards the estimated measurement matrix, and then the process is repeated. Although Huynh et al. only considered the case where the measurement matrix was complete, there is no reason the method could not also be applied when there is missing data. A similar outlier removal method has been applied as a part of the simulations below.

3.2.6 Comparison of methods

This subsection describes the results of some experiments with the standard dinosaur data set [11] available at http://www.robots.ox.ac.uk/$\sim$ vgg/data . Here, a video sequence

of 36 frames was taken of a toy dinosaur mounted on a turntable, which undergoes a full rotation over the course of the sequence. Harris corner points were then detected and tracked between frames, but due to the nature of the sequence there are many occlusions, and only about ten percent of the measurement matrix is filled. Figure 6 shows a picture of one frame of the dinosaur sequence, along with trajectories of the tracked points. Although a scaled orthographic camera model may not be strictly correct for this case, since the camera is relatively close to the model, it seems to be sufficiently accurate for a reasonable solution to be obtained. Although the actual coordinates of the individual points are not known, it is known that each of the points is moving in a circle, and so the feature measurements should lie on an ellipse.

The test on the dinosaur images measures the performance of various factorisation algorithms when some data is missing. Prior to applying the algorithms, those points that were only tracked into three or fewer separate frames were removed, leaving 1516 tracked points from the original 4983. Various methods and combinations of methods for determining the factors of the matrix were then applied to the data. The fit to the available measurements were summarised by two numbers: the root mean squared (RMS) error between the actual measurements and the reprojected model, and the median absolute error (MAE). The fit to the missing data can also be estimated by observing the trajectory of a point over the entire sequence, to see how far it deviates from a complete ellipse. In this case, the comparison was made for the feature point with the longest track, which appeared in 21 of the 36 frames.

Table 1 and Figure 7 compare the performance of the algorithms mentioned previously.

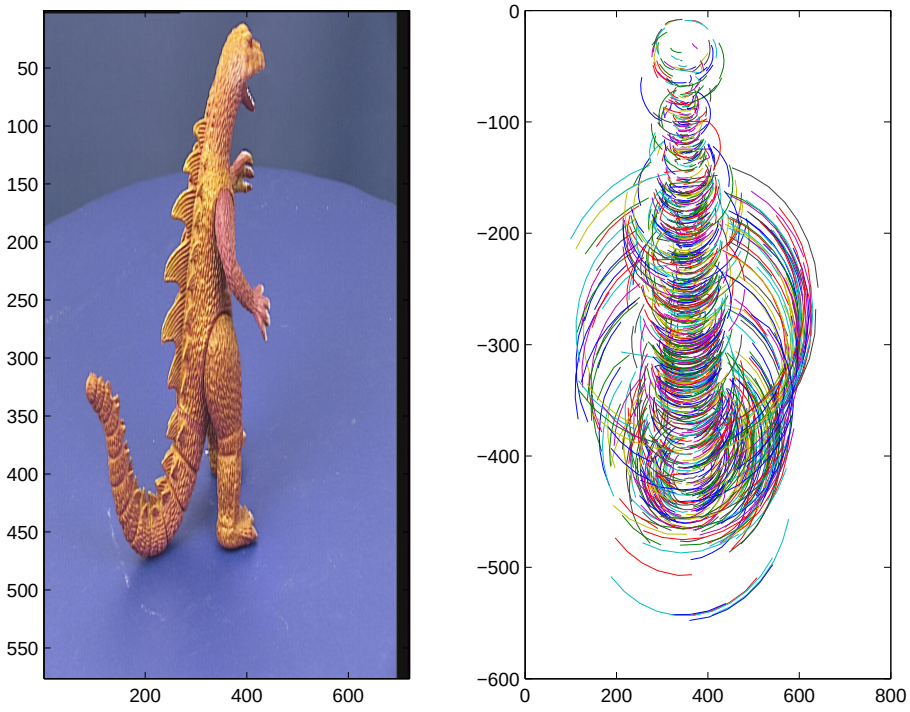

Figure 6: *Frame 1 of the dinosaur sequence, and the trajectories of tracked points.*

Method	RMS error	MAE
Jacobs	9.11	2.94
Tomasi and Kanade	5.90	1.40
Steepest descent	3.46	0.97
Iterated factorisation	2.08	0.45
Iterated least squares	2.07	0.45
Shum's method (160 iterations)	1.56	0.31
Shum's method (1000 iterations)	1.54	0.31

Table 1: *Comparison of measured and reprojected errors for some factorisation methods*

The numbers in the table are obtained by examining the difference between the reprojections of the models produced by each algorithm, and the original measurements for each track and frame of the sequence. The figure plots the measurements and reprojections for an individual point that has been tracked over roughly half of the image frames.

For the implementation of Jacobs method, only successive frames were used to produce the null-space matrix, otherwise the matrix was too large to factorise in MATLAB. While Jacobs method seems to work well for small matrices, it requires the factorisation of matrices much larger than the original, which makes it unsuitable for large amounts of data. The Tomasi and Kanade algorithms were used as starting points for the remaining algorithms. Iterated factorisation and least squares are only slightly different in effect, and in this case returned almost identical results with the latter method being much faster.

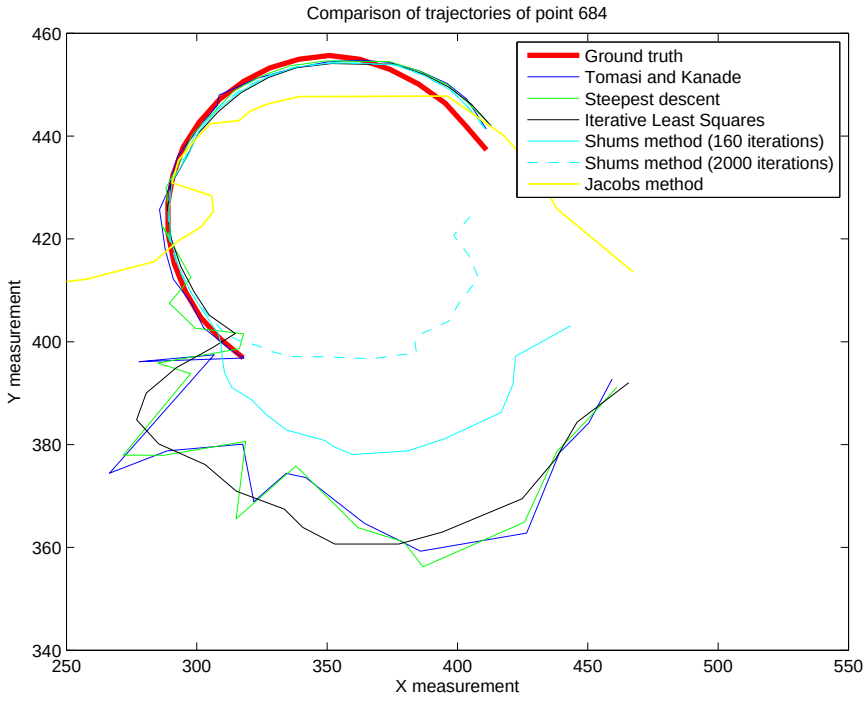

Figure 7: *Trajectories of the missing data from a tracked point using several methods.*

Shum's method worked best of those tested. There was still a significant change in the trajectories between 160 and 1000 iterations (as seen in the figure), despite the fact that the error decreased only marginally.

Table 2 and Figure 8 test some of the algorithms using an iterative outlier removal scheme. Here, the algorithm is run and the error between the measurement and reprojected model are compared. Those measurements with errors more than 4 standard deviations above the mean are then discarded, and the process repeated. This resulted in a substantial improvement in the point trajectories, and the MAE in all cases. In several instances the RMS error increased, which is likely due to the larger difference between the new estimate and incorrect measurements, which overwhelms the improvement in accuracy on the true measurements. Shum's method eventually converged to an almost closed curve, but this took thousands of iterations.

Method	RMS error	MAE	# removed
Tomasi and Kanade	5.26	1.08	104
Steepest descent	3.55	0.88	347
Iterated least squares	2.42	0.26	517
Shum (80 iterations)	2.57	0.21	396
Shum (800 iterations)	2.60	0.19	378

Table 2: *Comparison of measured and reprojected errors after outlier removal*

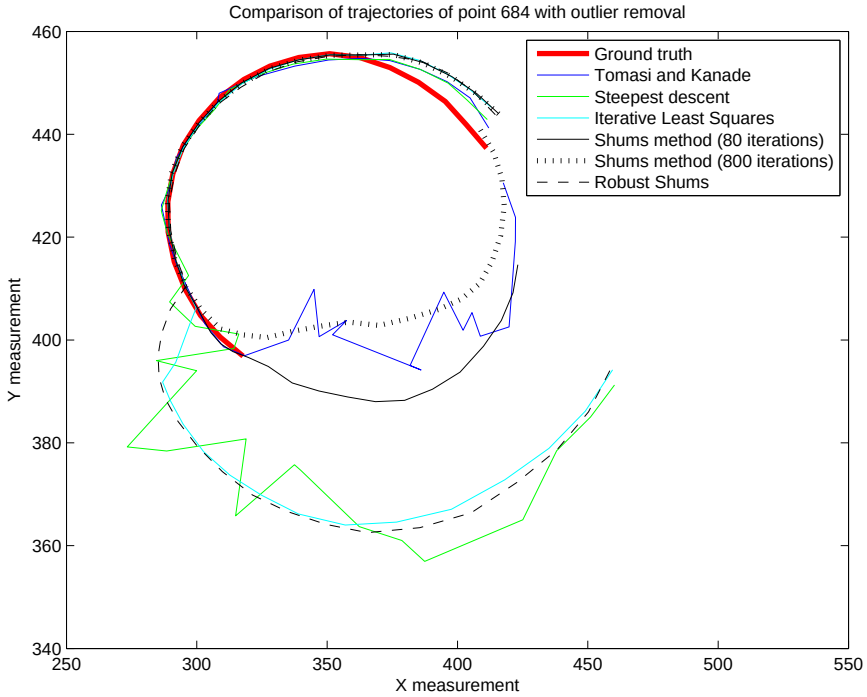

Figure 8: *Trajectories of missing data for various methods with outlier removal.*

Table 3 and Figure 9 describe another outlier removal method related to the scheme in [15]. Basically, instead of using linear least squares with an algorithm, an iterative weighted least squares can be used to produce a robust estimate (implemented as *robustfit* in the MATLAB statistics toolbox). This change has been made to both the Tomasi and Kanade hallucination algorithm, as well as Shum's method. Each combination of robust/non-robust initialisation or refinement is then compared as done previously. In general, the trajectories from the robust procedures out-performed the non-robust ones. However, the trajectory for the robust Shen method, when applied to the initial poor hallucination result, did not converge to a good result despite the low objective error measure. Instead, it seemed to split the data set into two halves, where there was a good agreement on the corners tracked in each half individually, but not on points that were in common to both halves. This suggests that there a small number of measurements that were important in achieving a correct solution, and these were effectively ignored by the robust weighting algorithm.

Refinement	Initialisation	RMS error	MAE
None	Non-robust	5.26	1.08
None	Robust	3.99	0.63
Non-robust	Non-robust	1.56	0.31
Non-robust	Robust	1.57	0.30
Robust	Non-robust	2.20	0.19
Robust	Robust	2.21	0.18

Table 3: *Comparison of robust with non-robust initialisation/refinement methods*

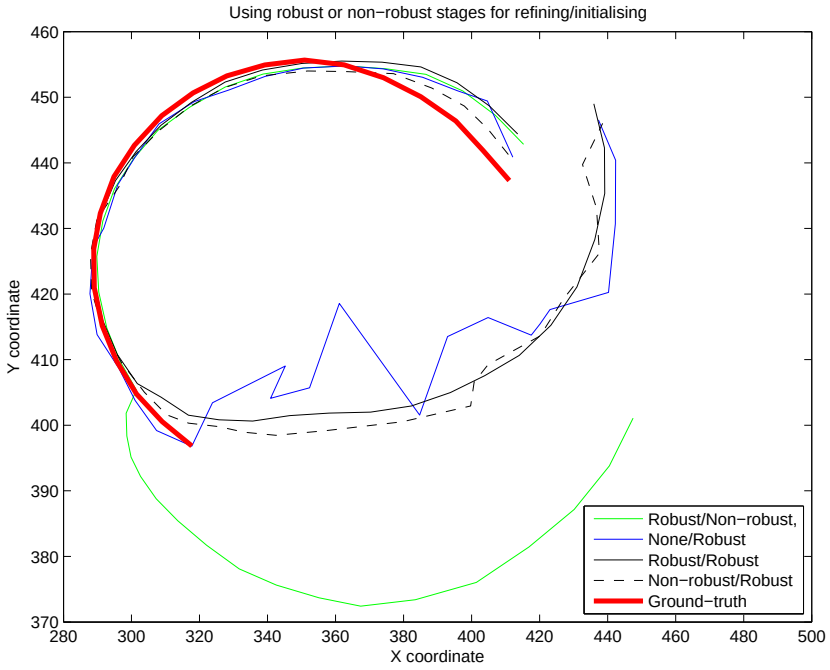

Figure 9: *Trajectories of missing data using robust/non-robust estimation.*

3.3 Track joining

In any given video sequence, there are likely to be some parts of the scene which are temporarily obscured or missing. Once features in these parts of the scene become visible again, they may become the start of a new track. To improve the accuracy of the resulting model, it would be useful to associate the new track with the old one. This has been implemented in the following naive way:

- **Hallucinate:** Use some sort of hallucination algorithm to fill in all of the missing elements in the measurement matrix.
- **Compare tracks:** Each column in the measurement matrix will now correspond to a different track. Every track should then be compared with every other to determine whether the two were at any stage measured in the same frame, and whether the predicted track from one feature was always within some threshold distance of the measured track of the other sequence. If the second property holds, but not the first, then the tracks should be joined, and the process repeated until no more consistent tracks are found.

The above track joining algorithm has been implemented in MATLAB, and applied to a sequence of tracked points generated using the software package Boujou. A subjective assessment indicated that it joined several of the longer tracks correctly, but no comprehensive measurements of detection and false match probabilities were made.

3.4 Line based factorisation

The previous factorisation algorithms have concerned measurements of point features. Quan and Kanade [22] have also extended the use of factorisation to detected line features. The resulting algorithm, however, is nowhere near as simple as for the line case.

The assumptions used for line based factorisation are as follows:

- A group of N_l lines have been measured in each image frame.
- The measurements of the ith line in the tth frame are characterised by a direction $\mathbf{u}_i(t) = (u_i(t), v_i(t))$ and a point on the line $\mathbf{a}_i(t)$. The point defined on each line does not necessarily directly correspond from frame to frame.
- The unknown parameters of the ith line are the 3D direction of the line $\mathbf{x}_i$ and a 3D translation $\mathbf{l}_i$.
- The camera uses an orthographic projection model.

The first stage in Quan and Kanade's algorithm is to estimate the camera rotation matrix and the 3D line directions $\mathbf{d}_i$ to be consistent with the measured 2D line directions $\mathbf{u_i}(t)$. This was solved using only three frames at a time, although how these frames should be selected from the available frames was not discussed. From the orthographic

assumption, the direction of the ith line in the tth image should satisfy $\mathbf{R(t)x_i} = \lambda_i(t)\mathbf{u}_i$, so that for three image frames

$$\begin{bmatrix} \mathbf{R} & \mathbf{u} & 0 & 0 \\ \mathbf{R}' & 0 & \mathbf{u}' & 0 \\ \mathbf{R}'' & 0 & 0 & \mathbf{u}'' \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ -\lambda \\ -\lambda' \\ -\lambda'' \end{bmatrix} = 0.$$

Since this equation must have a non-trivial solution, then the determinant of the matrix on the left hand side must be zero. This gives the tensor equation $T_{ijk}u_iu'_ju''_k = 0$ where the measured directions u, u' and u'' are known. There are seven unknowns (plus an overall scale factor which cannot be determined) in this equation, which means a solution for the tensor T (which is related to the camera rotation matrices) can be found with data from 7 lines. If all N_l lines are used, the system of equations to be solved becomes overdetermined, but can still be solved in a least squares sense using an SVD.

Once the tensor T has been obtained, it is still required to find the corresponding camera rotation matrices $\mathbf{R}, \mathbf{R}'$ and $\mathbf{R}''$. There are more degrees of freedom in the camera matrices than in the tensor T, which means that the camera matrices cannot be uniquely determined. This problem can be reduced by setting

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \text{ and } \mathbf{R}' = \begin{bmatrix} a_1 & \rho a_1 & -a_2 \\ a_2 & \rho a_2 & a_1 \end{bmatrix}.$$

Quan and Kanade then provide a set of complicated equations relating the unknown parameters to the tensor T. These provide two possible solutions for the camera matrices. The paper does not describe how to choose which is the correct solution. Once this is done however, the camera rotation matrices are still only determined up to an arbitrary projective transformation. This is because not all of the constraints implied by an orthographic projection have been applied. Subsection 3.1 described how to enforce the constraint that the x and y axes in the image plane should also be orthonormal in 3D space, and the same technique can be used here.

The final step is to find the translations of the lines in 3D so that they are consistent with the line positions in the images. Having determined the camera rotation matrices, each line in an image can be extended in the direction of the depth dimension to form an image plane. The intersection of the three image planes in space should then correspond to the position of the 3D line.

The above algorithm has been implemented in MATLAB for the Parafield fly-over sequence used in Section 2. The algorithm didn't appear to work, so to investigate why, the simulations given in the Quan and Kanade paper were repeated. The experiment involved three views of a cube, as shown in the first column of Figure 10. These figures were extracted from a PDF file of the paper. The end-points of the line segments were manually marked onto the image, with automatic post-processing (as described in Section 2.4) to align the line segment more closely with the actual edge. The factorisation based method was then applied to the line measurements. The resulting lines from the pentagon at the top of the box were then reprojected onto the original images. Since two possible solutions for the camera matrix are possible, these are shown in the first two diagrams

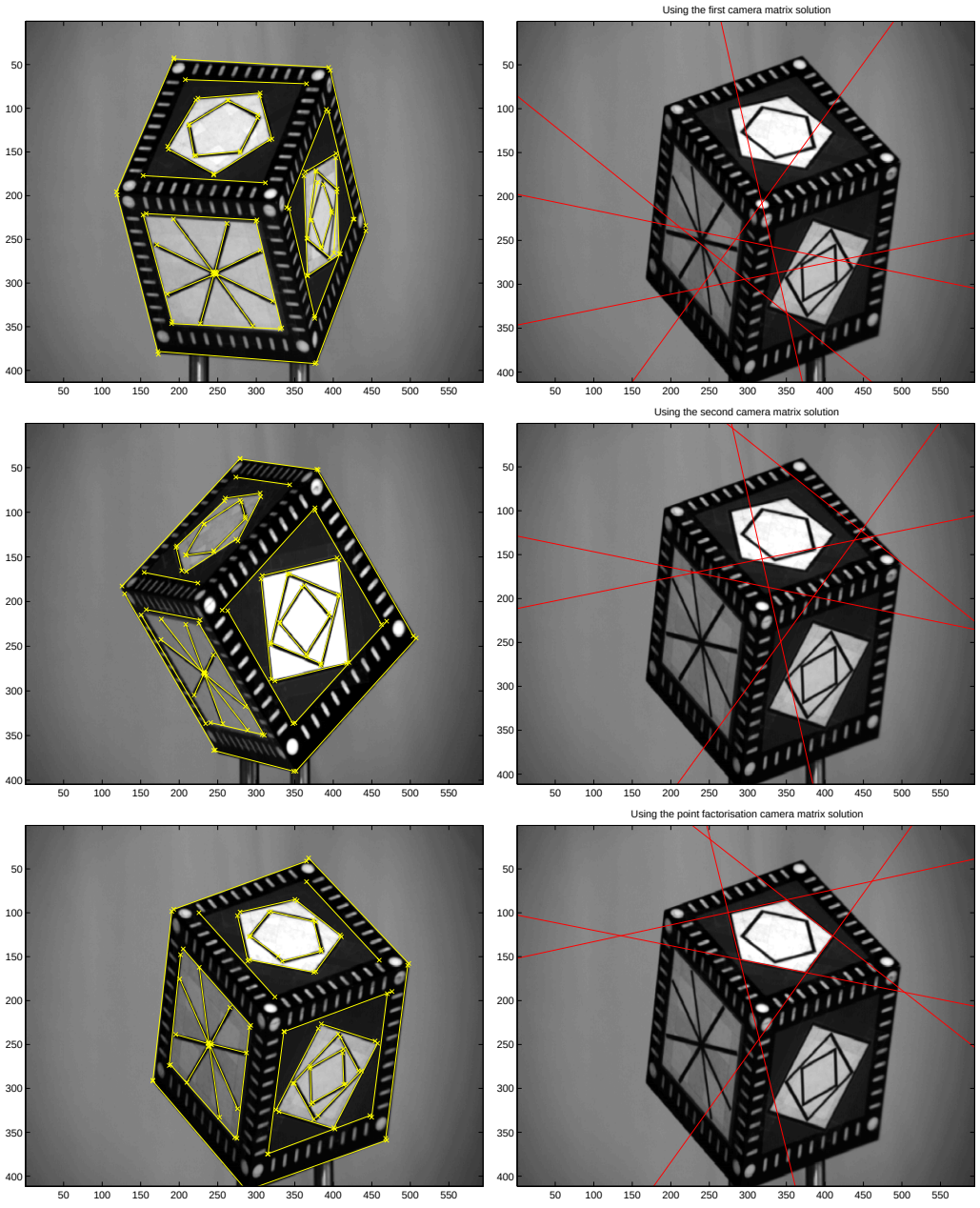

Figure 10: *Numerical results obtained using line based factorisation*

in the second column of Figure 10. The important thing to note from these diagrams is that the orientations of the reprojected lines seem quite accurate, but the translation is way off. This indicates that the first part of the algorithm is working as intended, but the resulting camera matrices are incompatible with the observed line translations.

To test whether the fault with the MATLAB code was the algorithm itself, the camera matrices were recomputed using a point based factorisation method. These estimates were then used in the second part of the algorithm to determine the line translations. The reprojections of the resulting lines from the pentagon are shown in the last diagram in the second column of Figure 10. This reprojection is much more accurate. Since the individual components of the line based factorisation appear to be behaving as expected, it seems as though it is the combined algorithm which is at fault. In summary, line based factorisation appears to be extremely sensitive to measurement error, despite the results reported by Quan and Kanade. It is not recommended that linear features be used alone as part of any factorisation method.

4 Post-processing

The structure from motion stage results in a set of camera poses for each of the image frames, and a set of points in 3D. When modelling a building, these points and any connecting lines, are theoretically all that would be necessary to produce a wire-frame model of the scene. In practice, however, not all of the important vertices will have been detected and tracked, superfluous points will be tracked, and individual points may have large errors in their estimated positions. This section describes two methods which attempt to deal with these problems. Subsection 4.1 attempts to reduce the effect of inaccurate points by perturbing the points closer to a model with planar facets. Following this, Subsection 4.2 tries to deal with the problem of superfluous points by segmenting groups of 3D points into planar regions. Neither method performed well, and are described mostly for completeness. A better approach seems to involve dense matching, where depth estimates are computed at every pixel within the image. Dense matching is described in more detail in Section 5.

4.1 Depth modification

The output of the factorisation method yields sets of points with noisy depth estimates. Some of this noise can be reduced by using additional knowledge of the structures being imaged, in this case buildings. Most buildings may be represented by a connected set of planar facets, and so if one could minimise the error between the existing set of points and this ideal model, the depth estimates could be improved. One such method for accomplishing this is described here.

The first step in the proposed scheme is to describe a metric for comparing the point model with the set of plausible building models. So that like is compared with like, the point model is converted into a surface model using a 2D Delaunay triangulation of the points as they appeared in one of the images (say, the first image of the sequence). It is then noted that if the points were to lie on an ideal model that the interior angles of

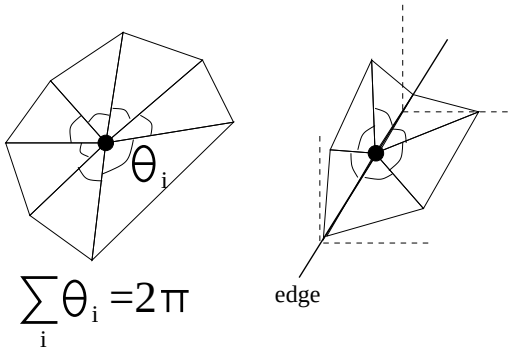

Figure 11: *Relation between interior angles for the facet model*

the triangles should behave in a consistent way at each point. This is shown in Figure 11 where it is easily seen that for a point on the interior of a facet, and for points on an edge, the sum of the interior angles of the triangles should be 2π. This is not correct for true corners but, in tests on the Parafield data with the Harris corner detector, it is evident that most of the tracked features are not true corners. Therefore, to produce a more accurate facet like model, one could modify the depth to locally minimise the following heuristic cost function

$$C = \sum_i \left(\sum_j \theta_{i,j} - 2\pi \right)^{2m}, \tag{2}$$

where $\theta_{i,j}$ is the interior angle of the jth triangle with a vertex at point i, and m is some positive integer. Of course, the global minimum to this cost will occur when all of the points lie on a flat plane, but local search techniques will tend to end up in local minima, which retain linear edges and planar facets, while mostly reducing spikes at the expense of rounding some corners.

The above function can be minimised using gradient descent or conjugate gradient methods. Both of these rely on finding the gradient of the function with respect to the dependent variable (*i.e.* the depth at each point). This can be calculated using the notation of Figure 12. It can be seen that when one of the corners is shifted by a small amount, represented by the vector $d\mathbf{n}$, then

$$\begin{aligned}
\cos(\theta_2 + d\theta_2) &= \frac{(\mathbf{a} + d\mathbf{z}).\mathbf{c}}{|\mathbf{a} + d\mathbf{z}||\mathbf{c}|} \\
&\approx \frac{((\mathbf{a} + d\mathbf{z}).\mathbf{c})|\mathbf{a}|}{(|\mathbf{a}|^2 + d\mathbf{z}.\mathbf{a})|\mathbf{c}|} \\
&\approx \frac{|\mathbf{a}|}{|\mathbf{c}|} \left(\frac{1}{|\mathbf{a}|^2} - \frac{d\mathbf{z}.\mathbf{a}}{|\mathbf{a}|^4} \right) (\mathbf{a}.(\mathbf{c} + d\mathbf{z})) \\
&\approx \cos\theta_2 - \frac{\mathbf{a}.\mathbf{c}}{|\mathbf{a}||\mathbf{c}|} \frac{d\mathbf{z}.\mathbf{a}}{|\mathbf{a}|^2} + \frac{d\mathbf{z}.\mathbf{c}}{|\mathbf{a}||\mathbf{c}|}
\end{aligned}$$

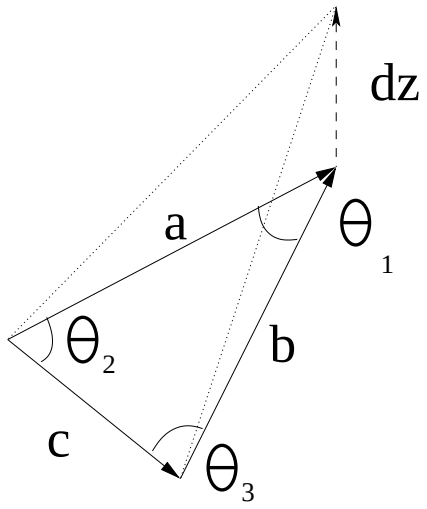

Figure 12: *The effect of changing depth on the interior angles of a triangular segment*

Also,

$$\cos(\theta_2 + d\theta_2) \approx \cos\theta_2 - \sin\theta_2 d\theta_2$$

from which, $d\theta_2/dz$ can be calculated when $d\mathbf{n} = [0, 0, dz]$, where the z-coordinate corresponds to the depth in the first image frame. The formula for $d\theta_3/dz$ can be obtained by substituting $\mathbf{b}$ for $\mathbf{a}$ in the above argument. Also, because $\theta_1 + \theta_2 + \theta_3 = \pi$ regardless of the point depths, then

$$\frac{d\theta_1}{dz} = -\left(\frac{d\theta_2}{dz} + \frac{d\theta_3}{dz}\right).$$

The gradient of the cost function can then be written as

$$\frac{\partial C}{\partial z_k} = 2m \sum_i \left(\sum_j \theta_{i,j} - 2\pi\right)^{2m-1} \sum_j \frac{\partial \theta_{i,j}}{\partial z_k},$$

which may be evaluated numerically using the above formulae. Using the Fletcher-Reeves update formula for the conjugate gradient method, the search direction d_k in the kth iteration of the numerical optimisation will be given by

$$d_k = \nabla C(z_k) + \beta_k d_{k-1}, \qquad \beta_k = \left(\frac{\nabla C(z_k)}{\nabla C(z_{k-1})}\right)^2.$$

One way to ensure convergence is to choose the step size to satisfy the Wolfe conditions, which state that if the new estimate of the optimum is $\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k \mathbf{d}_k$, then

$$\frac{C(\mathbf{z}_{k+1}) - C(\mathbf{z}_k)}{\alpha_k} \leq k_1 |\nabla C(z_k)| \qquad \text{Sufficient decrease condition}$$
$$|\nabla C(z_{k+1})| \geq k_2 |\nabla C(z_k)| \qquad \text{Curvature condition}$$

The processing described above, with $m = 2$, was applied to a set of points whose 3D positions were extracted using factorisation. Figure 13 shows before and after views of the models, seen from an angle where the differences between the models are most apparent. The two plots appear very similar, although the primary facets of the processed model do appear to be slightly better defined. The new points may be slightly easier to segment into plane regions than the previous set, although visually there is not a lot of difference between them.

4.2 Plane segmentation

The previous subsection described a method for changing the depth estimates so that the points were approximately aligned into planes. One way to form a dense reconstruction of these planes is to find which points belong to the same plane, and use this to construct a facet model of the building. The process of assigning points to planes is referred to here as plane segmentation.

Two methods for plane segmentation are considered in this subsection. The first is based on the 3D positions of points in space, while the second is based entirely in the image domain, and so does not require the use of factorisation.

4.2.1 3D position segmentation

The segmentation algorithm described in this section has the Koepfler algorithm [17] for image segmentation as its motivation. Like the Koepfler algorithm, the method presented here aims to minimise a global cost function. This function contains a measure of the goodness of fit of the model to the data, and a penalty for model complexity which is based on the Akaike Information Criterion (AIC). To obtain the cost function, it is assumed that the 3D coordinates of the ith point are (x_i, y_i, z_i) where all of the error is in the depth coordinate z_i. It is also assumed that the error is Gaussian with zero mean and a variance σ^2 to be determined later. Then if N points belong to a single plane, the mean square error in depth between the plane fitting the noisy data and the measurements will be $(N-3)\sigma^2$, where the value 3 is the number of degrees of freedom in choosing the parameters of the fitting plane. In the special case where only planes parallel to the ground-plane are to be detected (*i.e.* the tops of buildings), the number of degrees of freedom of each plane is reduced to 1, which should be used here instead of 3. For the current case however, a suitable cost function to minimise would be

$$C = \sum_{p=1}^{N_p} \sum_{i \in p} (z_i - \tilde{z}_i)^2 + 3\sigma^2 N_p,$$

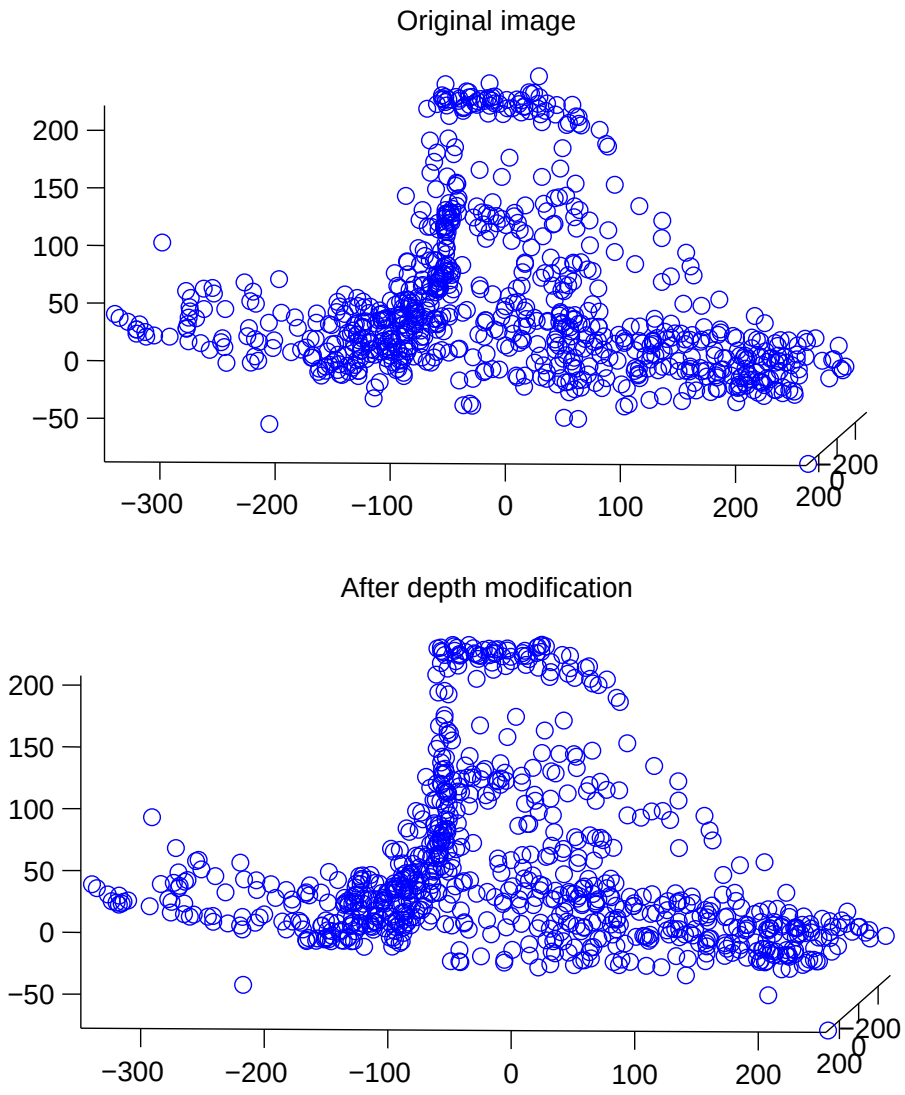

Figure 13: *The effect of depth modification on a 3D cloud of corners from the Parafield sequence*

where N_p is the number of points and $\tilde{z}_i$ is the estimate of the depth of the point i using the existing facet model. The value of the depth error σ^2 may be estimated from the imaging scenario, or may be measured from the 3D points themselves using the formula

$$\tilde{\sigma^2} = \frac{1}{\sum_i N_i} \sum_{p=1}^{N_p} \frac{N_i}{N_i - 3} \sum_{i \in p} (z_i - \tilde{z}_i)^2, \tag{3}$$

where N_i is the number of points assigned to the ith plane.

In order to assign the points to planes so that regions assigned to planes are continuous, a set of region mergings is performed (the coarse segmentation) followed by a fine

segmentation where individual triangles are swapped between plane areas.

The algorithm starts with an initial segmentation based on a Delaunay triangulation of the feature points. Here, every triangle corresponds to a separate planar facet. Since the error between the facet model and the measurements will always be zero in this case, it is not possible to obtain a useful variance estimate from equation (3) and another suitable guess should be made, until sufficient planes have been merged to allow a better estimate. For each plane, both the boundary points and the interior points are stored.

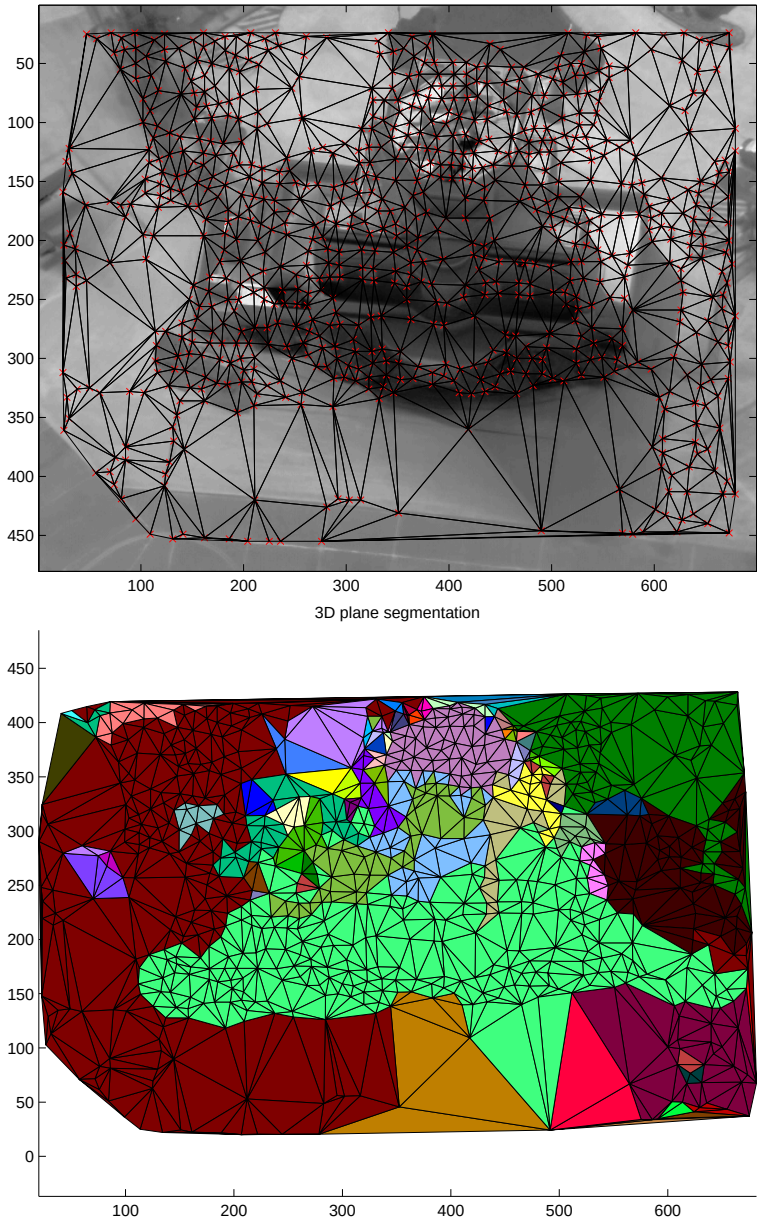

Figure 14: *The original image with triangulation, which have been coloured to indicate different segmented planar regions.*

The coarse segmentation, or region merging, is accomplished by calculating the effect on the cost function of merging every plane with every other plane which shares an edge. The minimisation is performed in a greedy manner by merging the two planes which give the greatest reduction in the cost function at each iteration, although this will not necessary lead to a global minimum in the cost. At each merge step, the boundary and interior points should be updated accordingly.

After the coarse segmentation, a fine segmentation occurs, where the cost of moving individual points near the edges of two planes from one to the other is calculated. Again, this point swapping is performed in a greedy fashion. Care should also be taken during this stage that a swap of a point should not result in points assigned to a plane becoming disconnected. After the fine segmentation, the process is repeated (coarse followed by fine segmentations) until no further change can occur to the segmentation without increasing the cost.

Figure 14 shows the result of applying the above plane segmentation technique to a 3D point cloud extracted from the Parafield fly-over. The top of the control tower has been reasonably segmented, while the ground-plane has too many segments, and the front of the building too few, resulting in a fairly poor model of the scene. Somewhat better segmentation can be achieved by tweaking the noise parameter σ, but there seems no simple method to choose this parameter in an automated way.

4.2.2 Affine transformation segmentation

Segmentation of an image area into planes can also be achieved by examining their relative motion throughout the image sequence. For an orthographic camera model, the image coordinates of sets of points on a plane will be related by an affine transformation. Different planes will correspond to different parameters of the affine transformation, which enables planes to be distinguished without explicit 3D information. The following is a method for accomplishing plane segmentation in this way:

- **Initialise:** Label all tracked points as "unassigned", which is used to describe points that do not adequately fit an existing plane.
- **Loop 1:** While there are still unassigned points, continue doing the following.
  - **Create a new plane:** Randomly assign half of the currently unassigned points a new plane label.
  - **Loop 2:** Repeat the following relabelling steps until the labelling becomes stable.
    - **Plane parameters:** Calculates the affine transformation parameters corresponding to each plane.
    - **Relabelling:** Each point is relabelled to belong to the same plane as one of its neighbours, based on the minimum distance to the plane affine transformations. Requiring a point to be in the same plane as one of its neighbours prevents isolated badly tracked points from being mislabelled. Points more than a certain distance from the plane may be labelled "unassigned" if one of its neighbours is.

In the above context, two points are defined as neighbours if their share an edge in the Delaunay triangulation of the points. This definition has a couple of advantages over the usual Euclidean neighbours. Firstly, if A is a neighbour of B, then B will also be a neighbour of A. Secondly, where the points are unevenly distributed, this definition ensures that the neighbours are more evenly distributed in angle, which helps to ensure scene continuity.

* **Cluster and merge:** All of the points assigned to the same plane are collected together to form clusters. The smallest cluster is merged into the closest of the other planes, or randomly split into the two closest planes if there is no consensus as to which is closest.

Figure 15 shows the result of applying eight iterations of the above algorithm to corners detected in the Parafield airport imagery. Each of the different coloured points indicate the different planes into which the points have been assigned. The black dots correspond to the currently unassigned points, and these are mostly clustered to the rear of the building, which is occluded over part of the sequence. As with the previous plane segmentation method, the ground plane is fairly well distinguished from the building, but the plane structure within the building itself is not especially useful. The blue points, for instance, actually do roughly lie within a plane, and appears to lie in a single cluster as each point in the region can be connected to any other with only links to neighbouring points belonging to the same region. In actuality, it consists of points from several different surfaces of the building, and so it is not a meaningful plane.

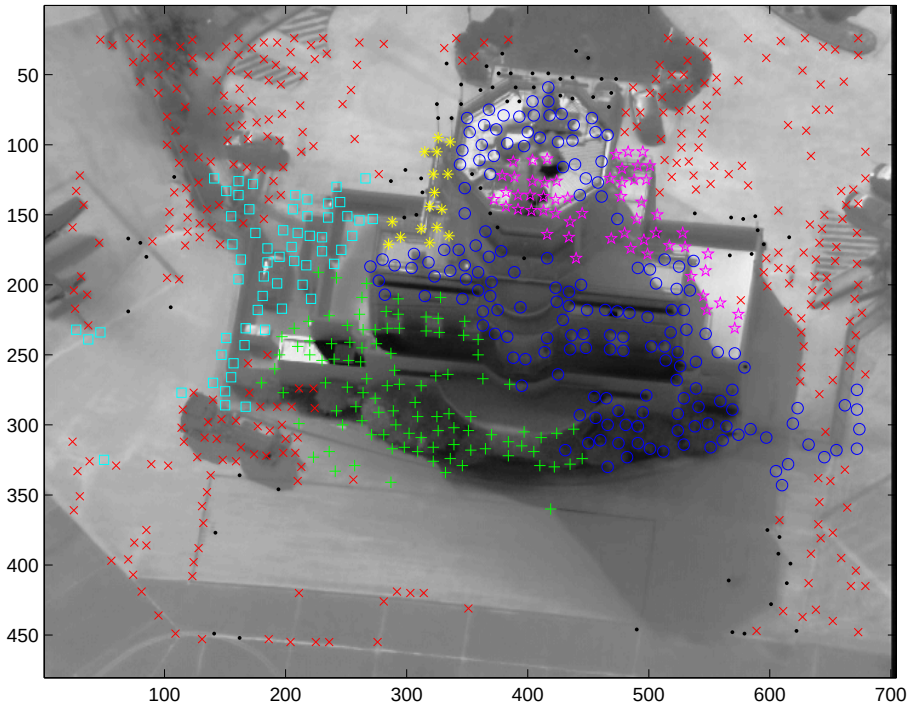

Figure 15: *Corner points automatically merged into separate planes based on affine transformations*

5 Dense matching

The first three sections of this report have described a procedure for extracting a sparse point model of a scene from a video sequence. The ultimate goal is to obtain a reprojected image of that scene from an alternative view, for which dense correspondences are required. The previous section has examined methods for accomplishing this based only on the sparse corner model. The current section examines related dense matching stereo techniques for obtaining 3D position estimates for every pixel in the image.

All stereo dense matching algorithms effectively measure optical flow between two images, with constraints to ensure that the resulting flow is consistent with a static 3D scene. In particular, points that project to a given image will project along a line in the second image called an epipolar line (and vice-versa). Any point projecting onto an epipolar line in one image must project onto a corresponding epipolar line in the other image. This means that if both images are transformed so that the epipolar lines are vertical [1], and they are scaled identically, then the optical flow must be vertical. The direction of the epipolar lines can be determined using the camera parameters obtained whilst obtaining the geolocated sparse model. Since a scaled orthographic model is being used, the epipolar lines will be parallel, but in the more general case the epipolar lines in an image will all meet at a single vanishing point. Figure 16 shows two frames from either end of a short video sequence of the Parafield control tower, which have been rotated so that the epipolar lines are vertical.

From the two frames in Figure 16, a dense 3D map can be constructed by finding how far each point in the image has moved between frames. This distance is known as the disparity. There are many algorithms available in the literature for solving this problem. Scharstein and Szeliski [27] have evaluated a large number of these using some pairs of images for which the disparity is known. More recent results on their benchmark data sets are available at the Middlebury Stereo Vision Page [28]. Although the evaluation function used by these two references seems consistent with the current application, the

[1] Most papers on dense matching assume the epipolar lines are horizontal

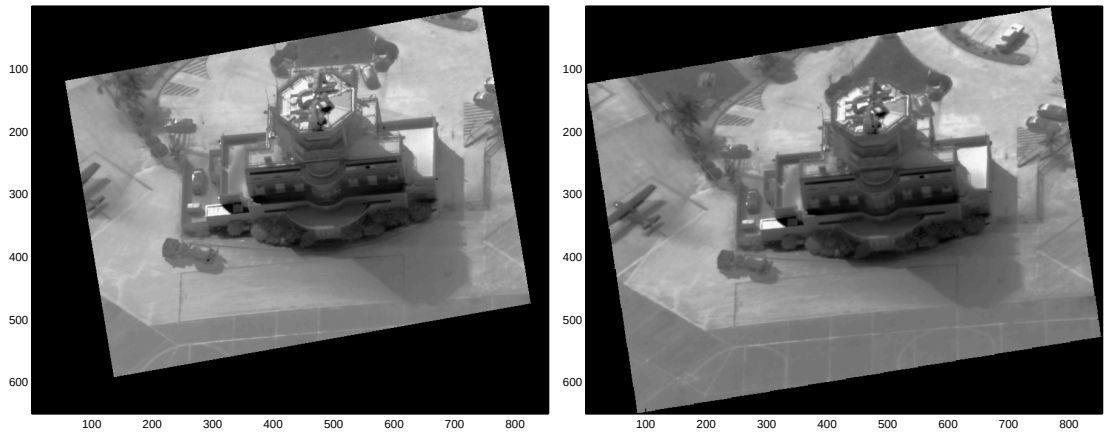

Figure 16: *Images from the Parafield airport sequence, rotated so that the epipolar lines are vertical*

test data they use to evaluate the algorithms is quite different in character. For a start, most of the regions are quite strongly textured, which allows accurate disparity estimates over large areas of the image. Secondly, their pairs of images are quite close together, resulting in small maximum values for the disparity, which are assumed known. This also means that there is little variation in intensity between the two images, allowing fast and accurate mean absolute difference of intensity metrics to measure the similarity of image patches. In the current application, the maximum disparity can be quite high, and will be effectively unknown, increasing the solution search space significantly. After a brief discussion of VRML in Subsection 5.1, Subsection 5.2 describes a new method for calculating disparity constraints, which can be applied to improve the solution of most matching algorithms. The remaining subsections describe the theory behind many of the dense matching algorithms described in the literature, as well as some attempted improvements. The Parafield MX20 data described in Section 2 has been used to compare some of these algorithms, and a more detailed account of this may be found in a separate DSTO report [9].

5.1 VRML display

VRML (Virtual Reality Mark-up Language) is a popular method for specifying a 3D scene (or "world") in a portable and consistent manner. Each world consists of a number of shapes, which may be collections of geometric primitives. Each shape can have its own position and orientation, and may be given its own texture, and material type (which affects how it reflects, refracts or emits light). The models generated in this section are based on only two camera views and can, in general, be adequately represented by an elevation model where each (x, y) position in the image can be associated with a single depth coordinate. In this case, the scene can be represented by a single primitive "ElevationGrid" which is defined in the following way

```
#VRML V2.0 utf8
  Shape {
    appearance Appearance {

      material Material {
        ambientIntensity 0
        diffuseColor  1.0 1.0 1.0
        emissiveColor 1.0 1.0 1.0
      }

      texture ImageTexture {
        url "im.jpg"
      }
    }

    geometry ElevationGrid {
      xDimension 163
      zDimension 214
```

```
    xSpacing   1.0
    zSpacing   1.0
    height [
            ......
           ]
    }
}
```

The above file specifies a single shape according to the VRML 2.0 specification. The appearance has been chosen so there is no ambient light because small variations in the surface texture were found to cause large shadows giving an unpleasant crinkly effect. Instead, the shadows (at least those generated by the VRML) are eliminated by assuming that the surface of the shape is emitting white light. The shape is also textured using the file "im.jpg" which is one of the frames used to construct the 3D model. The surface itself is of type 'ElevationGrid' with the specified dimensions, and the height profile (represented by dots in the above example) will be an array of numbers describing the height at each pixel location. The resulting VRML file can be examined on any VRML2.0 compatible viewer such as "Blaxxun Contract", which will allow the scene to be tilted, panned and zoomed to achieve the required view of the model. Most of the 3D models described in this report are screen shots from this viewer.

5.2 Interpolation

The standard stereo problem can only be solved by making some assumptions about the disparity map. In some cases, this is implicit (*i.e.* the object of interest lies completely within both images). More usual is to specify an upper limit on the disparity. The extra constraints on the solution generally allow a more accurate solution to be calculated much more quickly. This is also the reasoning behind hierarchical methods, where a rough disparity map is calculated at low resolutions, and is refined at higher resolution. Bobick and Intille [1] go further by choosing pairs of points within the imagery which they know to be the same (referred to as "ground points"), and choosing the disparity to enforce this. In our application, a large number of matches (corresponding to tracked corner points) are available, for which disparities can be calculated. However, a model based on linear interpolation of the disparity between the corner points will be quite jagged because some of the points have been poorly tracked. In order to deal with such inaccuracies, some extra slack in the disparities at these points must be allowed. The suggested heuristic approach is to use the corner points to adaptively find a likely disparity range for every point in the image. The likely range changes continuously between pixels, which enforces some measure of smoothness on the final solution, and can be obtained as follows:

- **Triangulate:** Construct a triangulation of the tracked corner points in one of the images. A better result will probably be achievable if the edges of the triangulation lie along edges in the image, but a standard Delaunay triangulation will do.
- **Estimate corner disparity range:** Because the corners have been tracked into the next frame, disparities d_i can be easily estimated at these points. Some of these

may have been poorly tracked, so the largest and smallest disparities of a corner and its neighbours (corners connected to it by the triangulation) are considered upper and lower values at the corner. Therefore

$$U_i = \overset{max}{t:i\in t} \ \overset{max}{j\in t} \ d_j$$

is the upper value, and a similar expression gives the lower value. In this expression, t denotes the points of a triangle, as generated by the above triangulation.

- **Interpolate:** Linear interpolation can then be used to obtain upper and lower values for the remaining points inside the triangles.

The above method has been used to generate a range of values for the dynamic programming dense matching approach, described in Subsection 5.4.

5.3 Cooperative stereo

An extremely simple way to measure the disparity of a point in the image is to take a small window around that point in the first image, and compare it to similar windows in the second image for all feasible disparities. The actual disparity would then correspond to the most similar match. In practice, this usually results in a great many mismatches, especially in regions with little discernible texture. To improve the smoothness and physical consistency of the resulting disparity map, Zitnick and Kanade [40] developed an iterative approach which they refer to as "cooperative stereo".

The cooperative stereo approach iteratively relabels points based on evidence both for and against a pixel having a particular disparity. Initially, a likeness measure $L_0(x, y, d)$ is calculated for a pixel at coordinates (x, y) having disparity d. Also defined is $L_n(x, y, d)$ which will be the direct evidence at iteration n that a point (x, y) has disparity d. Now the indirect evidence for a disparity of d at (x, y) will be that neighbouring row and column pixels are mapped to either the same point, or at least nearby points. This is the local support function defined by

$$S_n(x, y, d) = \sum_{r=x-N}^{x+N} \sum_{c=y-N}^{y+N} \sum_{D=d-N}^{d+N} L_{n-1}(r, c, D),$$

where $2N + 1$ is the window size, which was empirically chosen to be $N = 1$. For the purposes of this algorithm, the indirect evidence against a disparity of d at coordinates (x, y) is all of the evidence that a different coordinate is mapped to exactly the same point, or

$$R_n(x, y, d) = \sum_{c\neq y} S_n(x, c, y + d - c).$$

Any relabelling should be in favour of points with strong evidence of the correct disparity, and little evidence against it. The heuristic relabelling, given by

$$L_n(x,y,d) = \left(\frac{S_n(x,y,d)}{R_n(x,y,d)}\right)^{\alpha} L_0(x,y,d),$$

was chosen. Empirical results on some real images found the best value for α was about 2. The resulting disparity at each point can be found by choosing the value of d giving the largest value of $L_n(x,y,d)$ at each point. Due to the smoothing terms in the calculation of S_n, the resulting solution is much smoother than the original. Regions of the image belonging to occlusions tend to have a much smaller match score at the end, and so thresholding this quantity allows occlusions to be detected.

Figure 17 shows an example of the cooperative stereo algorithm applied to a white noise image, which has been shifted using a linearly varying disparity, with some additional independent white noise. The first figure shows that a number of false matches had been made based on the original correlation scores alone. After fifteen iterations, most of

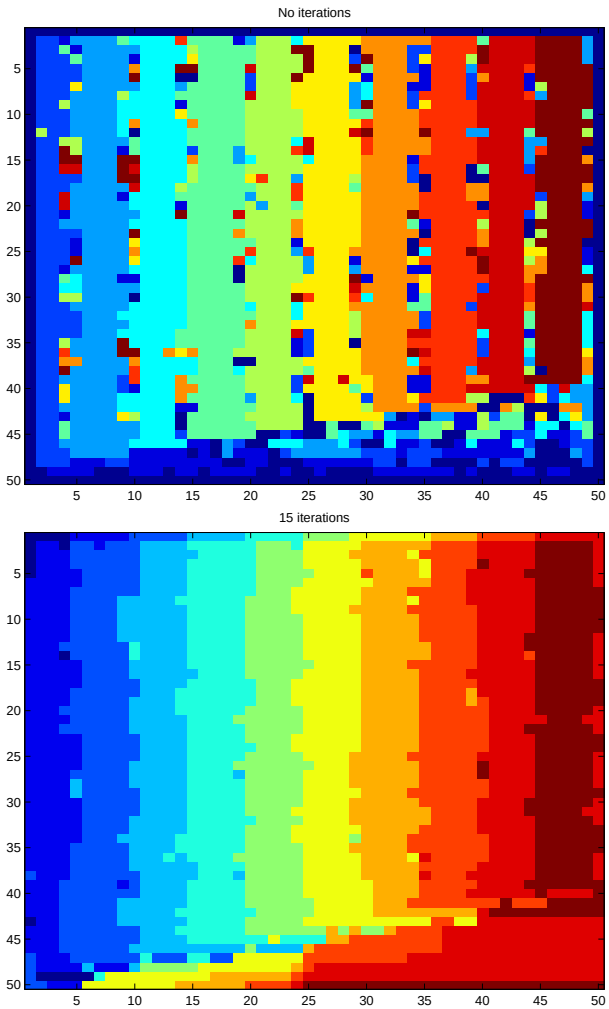

Figure 17: *The result of iterating the "cooperative stereo" algorithm applied to a linear disparity field*

these were removed. Scharstein and Szeliski [27] rank this algorithm somewhere in the middle of the pack for performance on their ground-truthed imagery. For our problem, the disparity range tends to be somewhat larger, and the inhibition region defined by R_n did not adequately constrain the disparity solution. The resulting map was not particularly suitable for registration purposes. A more detailed examination of the performance of the cooperative stereo algorithm for airborne imagery can be found as part of a separate report [9].

5.4 Dynamic programming

The dynamic programming approach [10] was, according to Scharstein and Szeliski, amongst the worst performing of all algorithms that they tested. However, airborne imagery has a number of features that make it in some senses more difficult than the examples considered by Scharstein and Szeliski. El-Mahassni and Cooke [9] have re-evaluated a number of these algorithms for airborne video, and found that many of the algorithms either break down completely, or are very parameter sensitive, and so are unsuited to automated tasks such as that considered here. The dynamic programming method appears to be a much more robust solution, and is also a lot faster than many of the competing methods, which should allow models to be computed in near real time.

The dynamic programming method for calculating disparity makes two basic assumptions. The first is that matches between pixels in corresponding epipolar lines (image columns) define a monotonic mapping. For relatively small baselines, this will generally be true, depending on the exact geometry of the scene. It is not necessarily fatal if it is not true, and usually no disparity measure at all (as for an occlusion) would be generated for the offending region. The second assumption is built into a cost function which is optimised over the entire image. The cost consists of a dissimilarity score between matched pixels and a term which penalises large changes in disparity between neighbouring pixels. The dissimilarity is given by $1 - c^2$ where c is the normalised cross-correlation between small regions about the matched pixels. This is used here instead of the more common sum of square or sum of absolute difference of image intensities because it is more robust to lighting and sensor variation. The discontinuity cost is defined to be zero for no change in disparity, a small penalty cost C_p for each single pixel change in disparity associated with a sloped surface, with an added cost C_o for a discontinuity, which is associated with an occlusion.

The cost function, described above, does not contain any dependence between epipolar lines. As a result, monotonic functions which minimise the cost function can be found for each line separately, although this does tend to produce some streakiness in the disparity estimate. For a given epipolar line, a length M column of pixels may be matched against another of length N, and the set of all possible mappings can be represented by paths through an $M \times N$ array.

The optimal function in the above array can be found using dynamic programming, as illustrated in Figure 18. Here, the minimum cost path from the left side of the array to the dark square is being considered. It is assumed that values for the cost have already been computed for all entries to the left, and the entries immediately above the currently considered point. Each of the previously computed values also contains a pointer back to

a previous entry along the minimum cost path. Now due to the monotonicity constraints, any path through the dark square must have passed through one of the squares on the stencil shown. If the scene is fairly flat at this point, then the minimum path should pass through the neighbouring diagonal square, and should not incur a penalty cost. If the scene has a prominent slope, then the minimum path will pass through one of the two neighbours, and incurs a slope cost C_s. If there is a discontinuity in the depth due to an occlusion, then the path will pass through one of the entries on the arms of the stencil, with a cost $C_o + C_p \Delta d$ where C_o, C_p are obscuration costs, and Δd is the size of the discontinuity. The minimum cost path back to the left hand side of the array will now have

$$C_{shaded} = C_{match} + \min_{i \in stencil} \{C_{disc.}(i) + C_i\}.$$

where the discontinuity cost C_{disc} is as described. After the array has been filled, the minimum cost in the final column will be the end point of the minimum path through the array, and using the pointers to work back through the array will yield the remaining points on the path. This can then easily be turned into a disparity.

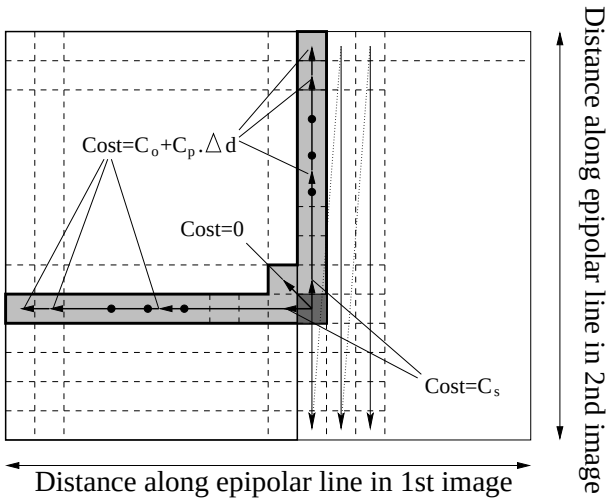

Figure 18: *A graphical depiction of dynamic programming matching for a single epipolar line*

The computation requirement in computing the minimum path in the above array can be reduced by limiting the paths to some feasible region. When fixed upper and lower bounds on the disparity are known, the feasible region will be a diagonal band through the array. Alternatively, upper and lower disparity values may be estimated from the corner points, as described in Subsection 5.2. In that case, the upper and lower curves of the feasible region will be piecewise linear.

The above algorithm was applied to the two images in Figure 16, and the results written into a VRML file. Two views of the VRML model are shown in Figure 19. From the plan view, the area behind the tower is shown in black because it cannot be seen in the first image, so no texture information is available here. Also, the 3D position estimates

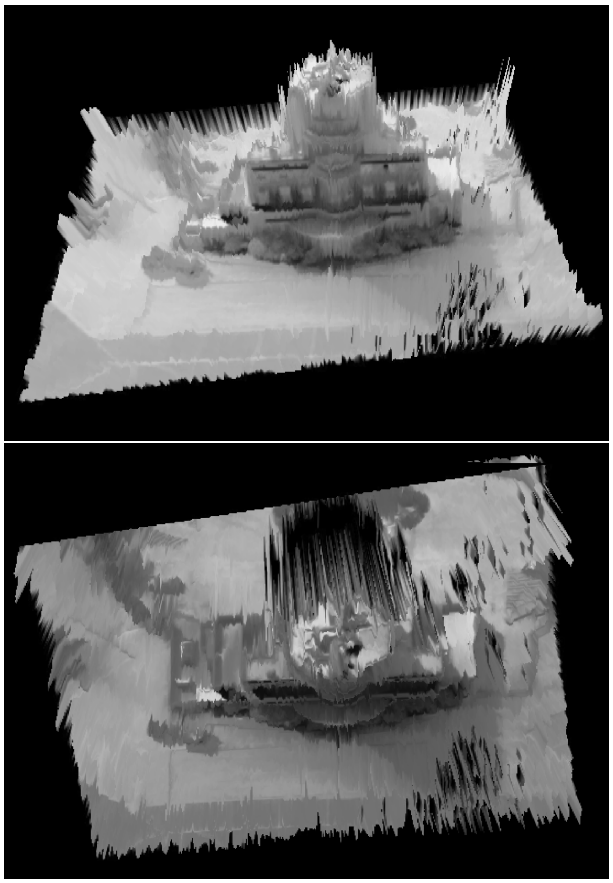

Figure 19: *Two views of the VRML model produced for the Parafield airport sequence using dynamic programming*

are still a little spiky, especially in the uniform regions near the edge of the image, for which good disparity estimates are difficult to obtain.

5.5 Graph-cut algorithms

A graph G consists of a number of nodes n and links (or arcs) l connecting the nodes together. The maximum flow problem assumes that there is a source and a sink node (s and f), and that each link l has a non-negative capacity $C(l)$, which limits the amount of "flow" that can be carried by that link. The problem is then to find the maximum amount of flow that the graph can carry from s to f. This problem was first answered by Ford and Fulkerson [12] in 1956. Their suggested solution was to find a path through the graph which was not at capacity, increase the flow along this path until it was at capacity, and then repeat until no further flow can be added. An illustration of this algorithm is shown in Figure 20.

The Ford and Fulkerson method (with an improvement by Edmonds and Karp) has a running time of $O(nl^2)$. A number of faster methods have since been developed. The push-relabel methods [13] are considered by a number of sources to be the fastest for most

ordinary graph problems. Boykov and Kolmogorov [3] have compared the two push-relabel methods (which they call H_PRF and Q_PRF), as well as the Dinic algorithm and a new method which they claim is faster for certain types of graphs. They state that although H_PRF is usually considered the faster of the two, Q_PRF seems to be faster for the types of graphs most frequently encountered in image processing. Since all of the programs used to obtain the results in this section were coded before reading this paper, the H_PRF method was used.

The last diagram of Figure 20 shows the graph at its maximum capacity. Here, every path from the source to the sink has at least one link that is saturated. In fact, these saturated links partition the graph in that a node could either receive extra flow from the source, or contribute extra flow to the sink, but not both (otherwise there would be a path from source to sink which was not at capacity, and so could accommodate extra flow). Any such partition of the nodes is referred to as a cut, since the links between nodes on opposite sides of the partition are effectively cut. The value of the cut is the sum of the capacities of the cut links. The partition defined by the maximum flow is in fact the minimum cut between the source and the sink.

The solution to the minimum graph-cuts problem (or the dual maximum flow problem)

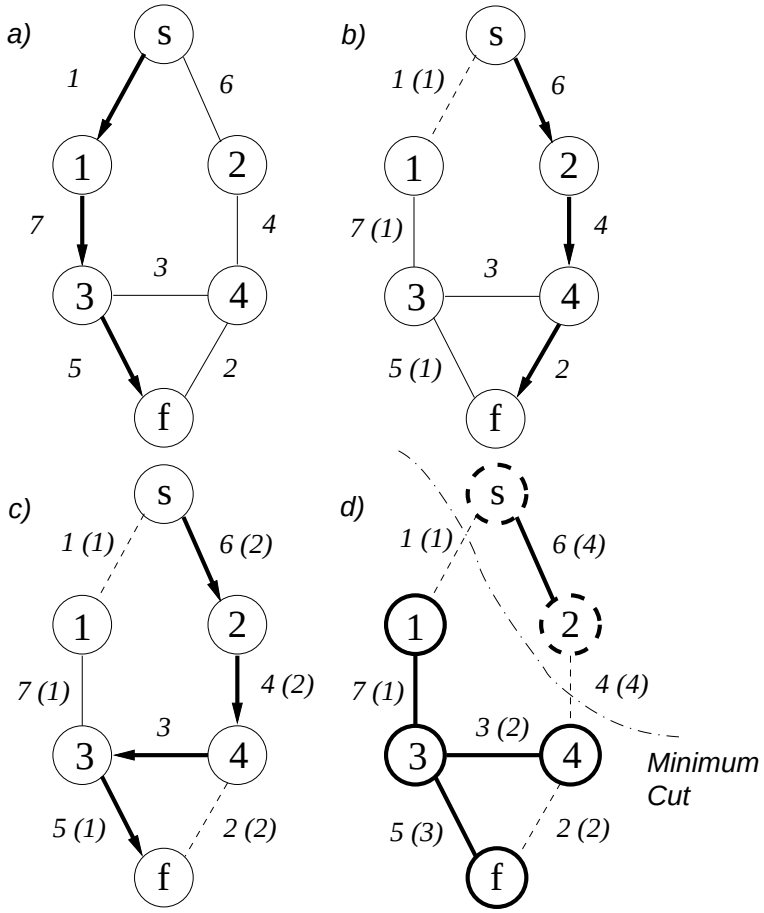

Figure 20: *Illustration of the Ford-Fulkerson algorithm applied to a graph*

has use in dense matching. The first application of this idea to dense matching appears to have been made by Roy and Cox [26]. They constructed a graph as shown in Figure 21, which they related to the dense matching problem by defining each column of nodes to correspond to a pixel, and each of the individual nodes to be a particular disparity for the pixel. A cut between the source and sink would intersect at each column at a position $d(i, j)$ which would be the estimated disparity at each point in the image.

The links along each column (i, j) were defined to have capacity $(C_(i, j, d) + C_(i, j, d + 1))/2$ where C is a matching cost function, because it was a "heuristic that works quite well in practice". They similarly defined the capacity for the links between columns, affecting the smoothness of the solution, to be $k(C_(i, j, d)+C_(i(+1), j(+1), d))/2$ for some smoothing constant k. The resulting dense map was apparently comparable to that obtained by dynamic programming, but removed the streaking between epipolar lines.

Another frequently cited graph-cuts paper is by Boykov, Veksler and Zabih [2] who related the capacities in the graph cuts problem directly to a global cost function. This allowed them to set up graphs to minimise a particular energy function. In particular, suppose a number of elements in some space (such as pixels in a picture, or voxels for a 3D space) are each to be labelled from a finite set $\mathbb{L}$, such as a set of disparities. Also suppose there is an energy functional over all possible labellings, consisting of a match term and a continuity term, which is to be minimised. This paper solved the problem by considering points currently labelled as $\alpha, \beta \in \mathbb{L}$ and setting up a graph so that the minimum cut

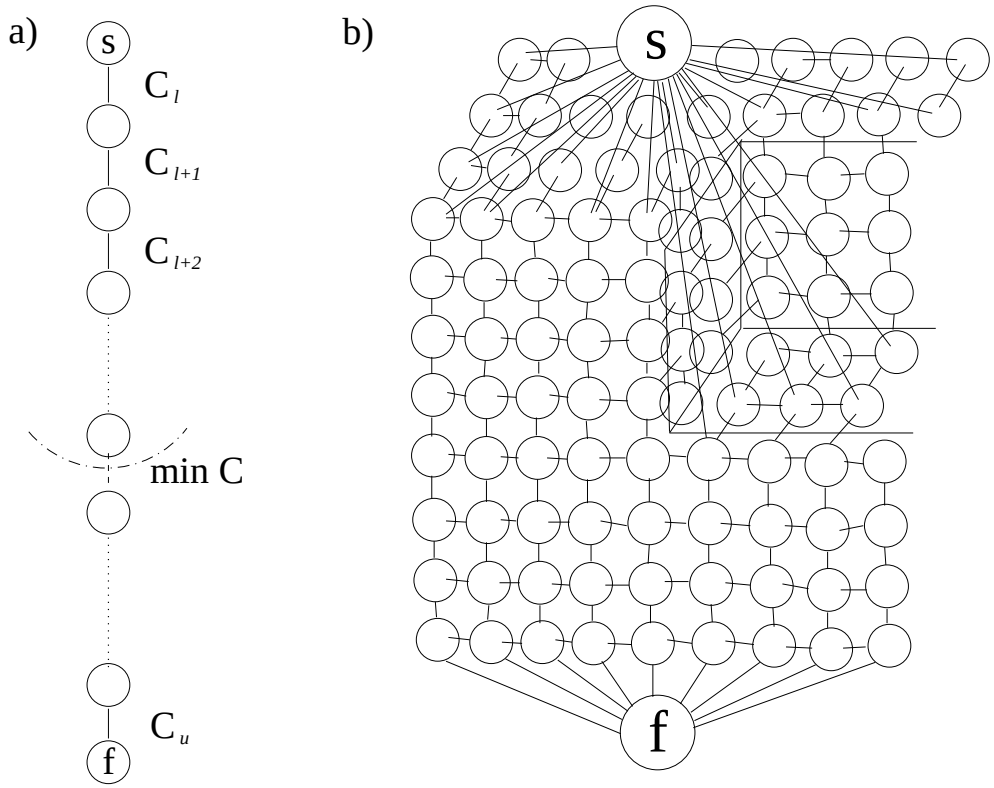

Figure 21: *Topology of the graph devised by Roy and Cox*

minimises the energy for these two labels. These $\alpha - \beta$ swaps are then repeated for all other pairs of labels, which the authors show will eventually lead to the optimal solution. The advantage of these $\alpha - \beta$ swaps over solving for all of the labels simultaneously (as in Roy and Cox [26]) is that a more general form of the energy functional can be minimised. The disadvantage is generally slower speed, especially when the number of labels is very high, as is the case for large displacement stereo problems.

An alternative to the Roy and Cox algorithm is now considered. The general topology of the graph is the same, with each column corresponding to a pixel. Since the minimum cut corresponds to cuts through links however, it makes more sense for each possible disparity to correspond to a link in the graph, instead of a node. Also, instead of a fixed minimum and maximum disparity for the entire image, different values may be used for each pixel. This will allow the interpolation procedure of Subsection 5.2 to be used as priors, and reduce the size of the graph to be solved. Alternatively, it would allow the graph-cuts to be applied hierarchically, with the solution at the lower resolution used to limit the bounds on the disparity at the higher resolutions.

For the link corresponding to a disparity d for the (i, j)th pixel, the capacity is defined to be the correlation based match cost $C(i, j, d)$ (in the real implementation, it is multiplied by 100 and rounded to the nearest integer, since the maximum flow algorithm only uses integers). This means that if there were no between-column links, the minimum cut would pass through the links having the best match for each column. The between column links are responsible for the continuity of the resulting disparity map, and were chosen to have a capacity k, which can be varied depending on the level of smoothness required. Some experimentation has found that when the correlation based cost is used, a fixed value of k can provide robust solutions for a number of images. Finding the minimum cut of this graph is the equivalent of minimising the energy given by

$$\min_{d(i,j)} \sum_i \sum_j C(i, j, d(i, j)) + k \left(|d(i + 1, j) - d(i, j)| + |d(i, j + 1) - d(i, j)| \right).$$

Ideally, one would like to have a modified energy term for jumps corresponding to obscuration. Although Boykov et al.'s $\alpha - \beta$ swap formulation would allow this, the extra computation time and complexity do not justify this approach.

Even using Boykov and Kolmogorov's code [3] to find the minimum cut in an n-vertex l-arc graph should take $\mathcal{O}(nl \log(n^2/l))$ [13]. When each image dimension is N pixels, the disparity range will be $\mathcal{O}(N)$ and so there will be $\mathcal{O}(N^3)$ vertices in the graph. As each vertex is connected only to its neighbours, the number of arcs will also be $\mathcal{O}(N^3)$ giving a total computation time of $\mathcal{O}(N^6 \log(N))$ which is prohibitive even for fairly small images. To reduce the computational requirements further, a hierarchical approach is taken. Hornung [14] does this by solving the full problem at a very low resolution, and then using this estimate to provide bounds on successively more detailed images of the scene. If the pixel (x, y) is found to have disparity d at the lowest resolution, then just limiting the disparity of the pixel $(2x, 2y)$ in the more detailed image to the range $[2d - 1, 2d + 1]$ gives poor results [36]. This is because the increased uncertainty within the image plane at the lower resolution has not been taken into account. Instead the range should be $[2d_{min} - 1, 2d_{max} + 1]$ where d_{min} and d_{max} are the minimum and maximum disparities

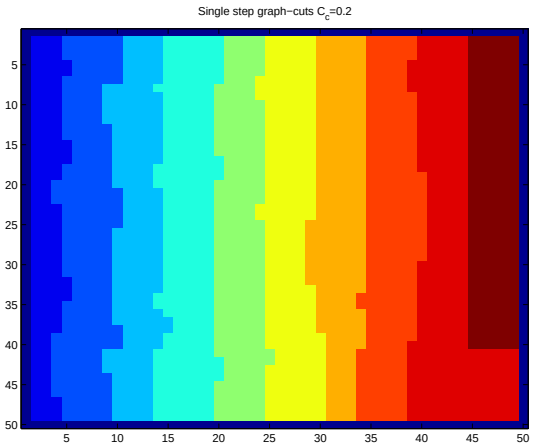

Figure 22: *Graph-cuts used to solve the toy example for a linear ramp*

over the the neighbourhood of the point at the lower resolution. The resulting hierarchical algorithm should have complexity $\mathcal{O}(N^4 \log^2(N))$.

Hornung's paper [14] also introduces several other novelties, such as the use of an octagonal grid to model both interior and exterior surfaces of the object, and the use of a cost function relating to multiple images. Very good results can be achieved without this, however. Figures 22 and 23 show results obtained using this graph-cuts algorithm.

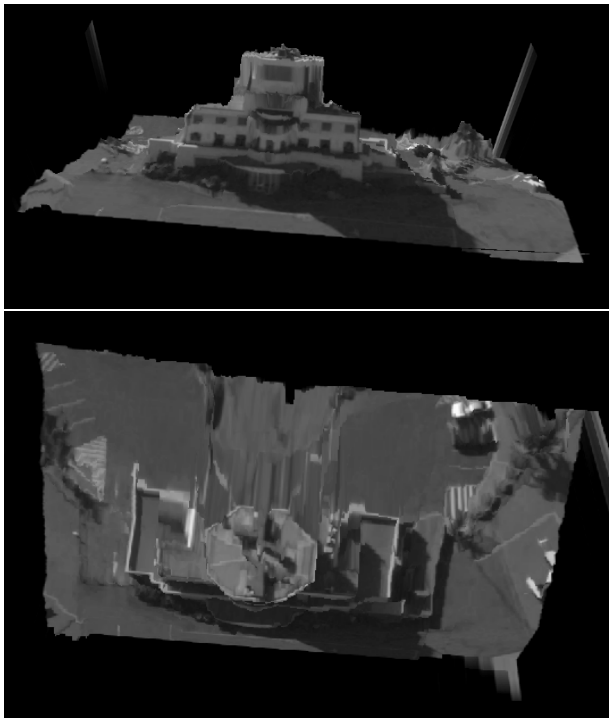

Figure 23: *Two views of the VRML model produced using graph-cuts.*

The first is for the same toy example generated for testing the cooperative algorithm in Subsection 5.3. It is evident that graph-cuts provided a more accurate disparity map in this case. The second example shows a 3D model extracted from an HDTV fly-over of Parafield control tower. This is the best-looking model of the control tower obtained so far. Although the original imagery is higher resolution than the MX20 data used to test the dynamic programming, the optical data has larger and more uniform areas than the MX20 images. As a result, when dynamic programming was applied to this same imagery, the resulting model was quite poor.

6 Error analysis

The process of registering airborne video to some reference imagery involves feature detection and tracking, structure from motion with missing data, georeferencing, dense matching and 2D image registration. This report has provided details on several of these areas, and a high level overview is available in a separate report [8]. The aim of the current section is to provide a framework for estimating the accuracy of the resulting registration, which considers the propagation of errors throughout each stage of processing. A description of the error estimation methodology is provided in Subsection 6.1. The quality of the estimated errors in some of the intermediate results have been checked numerically using an airborne video sequence of Parafield control tower, as described in Subsection 6.2.

6.1 Error estimation methodology

Figure 24 shows a flow diagram for the complete video registration problem with arrows, indicating processing methods, connecting measured or calculated quantities and their associated errors. The errors at the input to the process may be modelled by simple statistical models, but this becomes less and less appropriate further down the chain as the results depend more and more on larger scale structure within the image. At the end of the registration process, there may be a multi-modal distribution where the registration is either close to correct, or has been lured off by neighbouring similar looking buildings. This indicates that analytical methods for obtaining error estimates may be infeasible. Therefore, the approach to error analysis taken here is largely empirical. Another report [38] considering error analysis is currently in preparation and has some overlap with the analysis presented here [2].

6.1.1 Errors in feature detection

The errors produced in corner detection and tracking are probably the closest to being Gaussian i.i.d. noise of any component of the system. The exact distribution of the localisation error for each corner will be largely imagery dependent. There will also be a probability of an incorrect track, which will increase when there are neighbouring points in the image which appear similar in some sense. As there are not any useful models

[2] Comparing the two versions allows a meta error-analysis

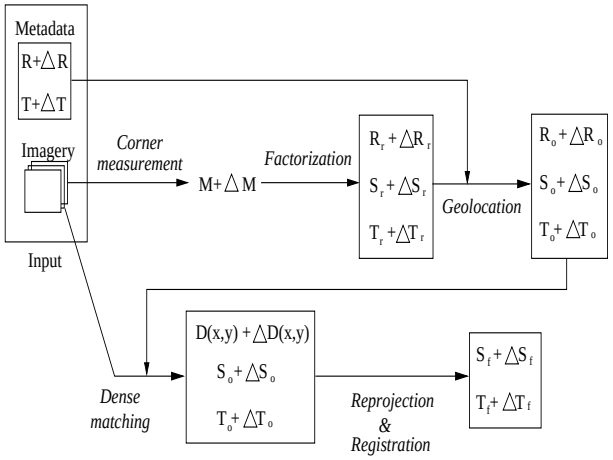

Figure 24: *Flow diagram showing the errors throughout the video registration process*

for our background imagery, a pragmatic approach has been taken and the errors are estimated from the difference between the measurement matrix M and the reprojected model $M_r = R_r S_r + T_r$. A theoretical expression for this error is available under certain special assumptions. Assume that the scaled orthographic assumption is accurate, the measurement noise is i.i.d. Gaussian with variance σ^2, the matrix is completely filled, and the translation is zero. Chen and Suter [5] showed that in this case, the mean absolute difference between the actual matrix and the reprojection is

$$E(|M_{ij} - M_{r,ij}|) = \sigma\sqrt{\frac{r(m+n) - r^2}{mn}}, \tag{4}$$

where $m \times n$ is the size of the measurement matrix, and r is the rank of the measurement matrix. The $r(m+n)-r^2$ in the numerator is the number of degrees of freedom in choosing the factors (an $m \times r$ matrix, an $r \times n$ matrix, and a set of ambiguities corresponding to an arbitrary $r \times r$ matrix). The denominator mn is the total number of measurements in the complete matrix. They then hypothesised that in the case of incomplete data, when a fraction ρ of the measurements is missing, the mn is replaced by ρmn. The estimate suggested here incorporates this factor, and also tries to take into account the ease with which features are tracked by estimating a separate value of the noise σ for each of the tracked points.

6.1.2 Factorisation and geolocation error

Following the detection and tracking of corner points over multiple frames, the next step is to estimate the 3D positions of these corner points, and the camera poses. As this step has no explicit dependence on the imagery, it is the most amenable to a mathematical treatment of the errors. Some of this analysis is now provided for some simple scene and camera geometries, along with some explanation of some of the other error effects which have not yet been modelled. This is then followed by a description of a Monte-Carlo

method for evaluating the errors in a more realistic scenario. A numerical example showing the effectiveness of this Monte-Carlo approach is given later in Subsection 6.2.

A Simple Scenario

Consider two parallel line segments of length l which are perpendicular to the optical axis of the camera, but lie on a plane containing the optical axis. Suppose the first line is a distance D from the camera, and appears to be N pixels long in the image (corresponding to a scaling factor of $s = N/l$). The second object is $D + d$ from the camera. In this scenario, there are a number of different types of error associated with extracting 3D models of the scene. These have been addressed below roughly in order of the scale at which the errors occur. For instance, the first errors affect every point in the model, while the last errors will affect the solution only on a feature by feature basis.

Georeference location error: The factorisation and bundle adjustment algorithms discussed previously may be useful in determining the relative positions of feature points. They only provide information about the actual position of points in space up to a translation and rotation. These values must be determined from some other source, such as meta-data.

Perspective error: This effectively refers to the model error, or the error produced by using simplifying assumptions about the camera and scene (*e.g.* the scaled orthographic assumption) to construct the 3D model. In the scaled orthographic camera model, the two line segments in the example would appear to be N pixels long. In the more accurate perspective camera model, the apparent size would be $ND/(D+d)$. The difference between the camera models will be consistently measurable when it is larger than the pixel size, so that

$$N\frac{d}{D+d} > 1.$$

Overall scale error: In the scaled orthographic model, there is no way of distinguishing between the amount of zoom (the ratio of focal length in pixels to target distance in external units) and the scale of the target. The only way this information can be obtained is from the associated meta-data. Alternatively, a perspective model could be used, but here the scale estimate will only prove accurate in situations where there is significant disagreement between the orthographic and perspective models.

Structure/motion ambiguity: Assuming that the overall scale has been determined, there is still some difficulty in estimating the scale of the structure in depth. The case where a second view of the the scene is available is now considered. In practice, many intermediate frames will also be available, but it is not altogether clear how helpful these will be. A common assumption is that the errors in the positions of corner points in different frames are i.i.d Gaussian, and so the error should vary as $1/\sqrt{N_f}$ where N_f is

the number of frames. However this model is flawed in that it predicts that the resolution of a static camera viewing a static scene can be improved by collecting frames for long enough. The model ignores temporal correlation of the errors. Furthermore, if N_c is the number of corner points in the measurement matrix, Equation (4) from Subsection 6.1.1 indicates that the addition of more points is not very helpful. This is because as $N_c \to \infty$, the mean absolute difference between the rank-3 measurement matrix and the model reprojection tends to $\sigma\sqrt{3/(2N_f)}$, which is independent of the number of points (Note that $m = 2N_f$ and $n = N_c$). While the addition of further points and frames probably does improve the error in the resulting model, it may not be by as much as might be supposed, and consideration of the two frame case will at least provide an upper error limit.

Suppose that only one other view of the scene is available, where the camera is rotated in the plane containing the lines and the optical axis by an angle θ, as shown in Figure 25. For the scaled orthographic case, both lines will appear to be the same length, but with a relative translation (in pixels) of d_r. When there is no relative scaling, this will be related to the other parameters by the equation

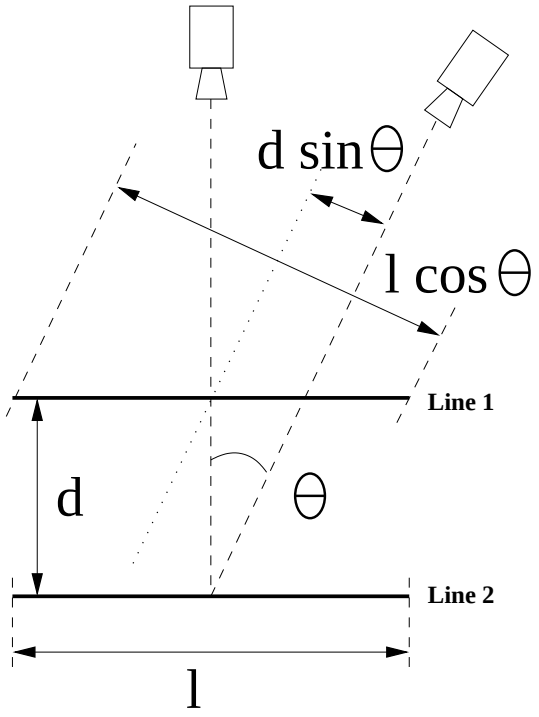

Figure 25: *Geometry for determining the depth scale ambiguity*

$$d = \frac{d_r l}{N \sin\theta}. \tag{5}$$

The translation d_r will appear exactly the same as if the camera had rotated through an angle of θ_2 and the distance between the lines was $d \tan\theta / \tan\theta_2$. Unless meta-data

for the camera angle is available, the only way to discriminate between the two cases is to use the change in the length of each line. The camera angle can be estimated from the changed length N_c (scaled to account for image magnification) according to the formula

$$\theta = \arccos\left(\frac{N_c}{N}\right).$$

Substituting into equation (5) gives the depth as

$$d = \frac{d_r l}{\sqrt{N^2 - N_c^2}} = \frac{d_r}{s}\frac{N}{\sqrt{N^2 - N_c^2}}.$$

From this, it can now be determined how the depth d of the scene could change, and still give estimates consistent with observations of the quantities N, N_c and d_r. The allowed variation in depth may be obtained from the linearised error formula

$$d + \Delta d = \frac{d_r}{s}\frac{N}{\sqrt{N^2 - N_c^2}} + \frac{\partial d}{\partial d_r}\Delta d_r + \frac{\partial d}{\partial N}\Delta N + \frac{\partial d}{\partial N_c}\Delta N_c$$

where

$$\begin{aligned}
\frac{\partial d}{\partial d_r} &= \frac{N}{s\sqrt{N^2 - N_c^2}} \\
\frac{\partial d}{\partial N} &= \frac{-N_c^2 d_r}{s(N^2 - N_c^2)^{3/2}} \\
\frac{\partial d}{\partial N_c} &= \frac{N_c N d_r}{s(N^2 - N_c^2)^{3/2}}
\end{aligned}$$

The values of N, N_c and d_r must be measured from the imagery, but due to discretisation they cannot be known exactly. These values are measured pixel lengths between the end points of line segments, whose position will be unknown up to half a pixel (or $1/\sqrt{2}$ pixels if the lines appear in the camera images at 45^o to the image axes). Therefore, in the worst case, the errors in the measured pixel distances will be $\sqrt{2}$ pixels. There may also be an additional localisation error due to detection and tracking of the corner points, but this has not been considered here. Also, the discretisation error of N_c will be different if there is any zoom between the initial and final frames. If the objects in the final frame have been reduced by a known factor α (*i.e.* increasing α means the lines appear shorter in the second image), then the resulting depth error will be

$$d \pm \Delta d \approx d\left(1 \pm \sqrt{2}\left(\frac{1}{d_r} + \frac{\alpha N_c + (N_c^2/N)}{N^2 - N_c^2}\right)\right) \tag{6}$$

in the worst case when all errors make changes of the same sign. When meta-data is available, the change in the camera orientation between the initial and final frames need not be

estimated using factorisation, and the change in depth consistent with the measurements may be obtained directly from equation (5) to give

$$d \pm \Delta d \approx d\left(1 \pm \left(\frac{\sqrt{2}}{d_r} + \frac{|\cos\theta|}{|\sin\theta|}\Delta\theta\right)\right). \tag{7}$$

in the worst case, when all errors make changes of the same sign.

Quantisation error: There are two types of quantisation error which will manifest themselves in the final 3D feature model. The first is azimuth error, whose value in radians is up to half a pixel size divided by the focal length. The second is in depth. Suppose that the depth scale factor, produced by the structure/motion ambiguity, has been calculated correctly. Then there will still be errors in calculating the relative depth between two points due to pixelation error. The actual projected motion of two points differing in depth d will be $d\sin\theta$. If the last frame is zoomed out by a factor of α compared with the first frame, then the two points can be resolved only if the relative shift is larger than half a pixel. Rearranging this gives the minimum resolvable depth to be

$$d = \frac{(1+\alpha)l}{2N\sin\theta}$$

Centre of mass error: One of the assumptions of the factorisation algorithm is that the coordinates of the points in the image are known with respect to some fixed point. In reality, no such point is available, and the centre of mass of the tracked corner points is used. There will be an error in estimating the position of this point, which will further lead to an error in the resulting estimated positions. This error is, however, believed to be very small but difficult to quantify, and so has not been analyzed.

Monte-Carlo Simulation

The simple scenario just addressed considered two frames, and two points. In practical cases, a sparsely filled measurement matrix will be available, and the overall model will combine the results from multiple frames and many tracked points, each of which may be assigned a different weighting depending on the method used for dealing with the missing data. The errors in the measurements due to quantisation are also not likely to be completely independent. For instance, two measurements in consecutive frames (for a sufficiently high frame rate) will have correlated errors, as will two features that are close together spatially. For this reason, it is difficult to produce an accurate error measurement from the simple scenario. One method that has been found to work well is the Monte-Carlo approach.

The approach shown in Figure 26 produces an estimate for the error in 3D structure and camera poses as produced by factorisation followed by georeferencing. The georeferencing step is required because factorisation produces estimates for the camera poses and

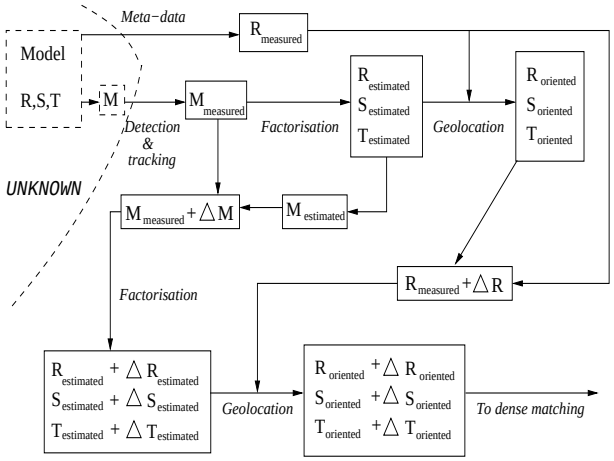

Figure 26: *Flow diagram of the error estimation process*

structure that still has ambiguities. For instance, the camera and the scene may both be given the same overall rotation without affecting the measurements. As a result, additional information in the form of metadata is required to produce a correctly oriented model. This metadata typically contains information about the position and orientation of the platform and the sensor, from which an estimate for the camera matrix, R_M, can be obtained for some subset of video frames. The geolocation stage uses linear least squares to find the mapping which best aligns the matrix R produced from factorisation, with the metadata R_M, resulting in a correctly oriented set of camera poses R_o and 3D feature positions S_o. That is, a matrix T is found so that

$$\tilde{R}T = \tilde{R}_M, \quad R_o = RT \quad S_o = T^{-1}S$$

where the tilde indicates that rows of the matrix, for which meta-data is not available, has been removed. A more complete description of the use of meta-data in structure from motion is provided by Whatmough [38].

When the errors in the metadata measurements are well understood, then ΔR_M can be determined relatively easily. More generally, these errors may require estimation as well. This is accomplished here by comparing the metadata with the oriented camera poses. The error ΔR is decomposed into a zoom or radial error, which is the root mean squared difference in the magnitude of the camera parameters, and an angular error which considers the r.m.s. difference between the theoretical and measured orientations of the camera axes.

The factorisation and geolocation stages produce what should be a correctly oriented set of 3D feature positions within the scene. Having estimated the errors in the feature measurements and the metadata, some measure for the error in these 3D points may now be determined. This is done here by producing a hundred new measurement matrices and metadata camera pose matrices by perturbing the existing measurements. The measurement matrix M is perturbed using i.i.d. Gaussian noise followed by discretisation, with a different variance for each of the tracked points, as mentioned earlier. Similarly,

new camera metadata is generated by perturbing the zoom and the angle of the observed metadata. The factorisation and geolocation steps are then applied to this random data to generate an ensemble of point clouds, from which the errors in a given 3D feature point may be determined.

6.1.3 Error in dense matching

The dense matching step uses the estimated camera poses to calculate the disparity between two widely separated images for every pixel in the image. The suggested method uses graph-cuts to optimise a global cost function consisting of a match cost between pixels in each image and a measure of the discontinuity of the scene. As a result of the strong dependence on the imagery and the scene, for which there are no general models, this stage does not seem amenable to any form of error analysis. Therefore, a heuristic for incorporating the error from this stage is to extend the range of the errors in depth until it encloses the dense match surface.

6.1.4 Error in registration

The final stage of the process is registration. At the moment, work on 2D registration against reference images and 3D registration against CAD models is still in progress. Commenting on the errors at this point is therefore perhaps premature. It is expected that an operator of the system would only really be interested in two errors. The first is the probability that the match is completely wrong, and the second is that if the match is correct, what is the likely error in the registration of the points. If 2D registration is performed using something like a correlation which assigns an overall score to the match, it should be possible to apply several scaling and rotation perturbations (with size relating to the error in camera pose) to the dense model and hence calculate the error in the registration score. The variability of the maximum match score, along with the match score in the area around the best match position should provide some information about the error in registration. The likelihood of a complete mismatch would be related to the number of peaks in the match score that are close to the maximum. Similarly, if the match is roughly correct, the translation error should be related to the sharpness of the maximum, as measured by the second derivative of the match score.

6.2 Numerical example

This example tests the ability to estimate errors in estimates for the 3D positions of points using the Monte-Carlo approach described in the previous subsection. The data used was extracted from 60 seconds of an HDTV video sequence showing Parafield airport control tower, in South Australia. An example frame from the sequence is shown in Figure 27a. No usable metadata was available with this particular sequence.

The standard KLT tracker was used to detect and track feature points between the frames, and then the frames were subsampled to give one frame per second. A total of 2079 features were tracked through at least four of these subsampled frames. Because the trajectory of the camera with respect to the object was a lot less variable than for the

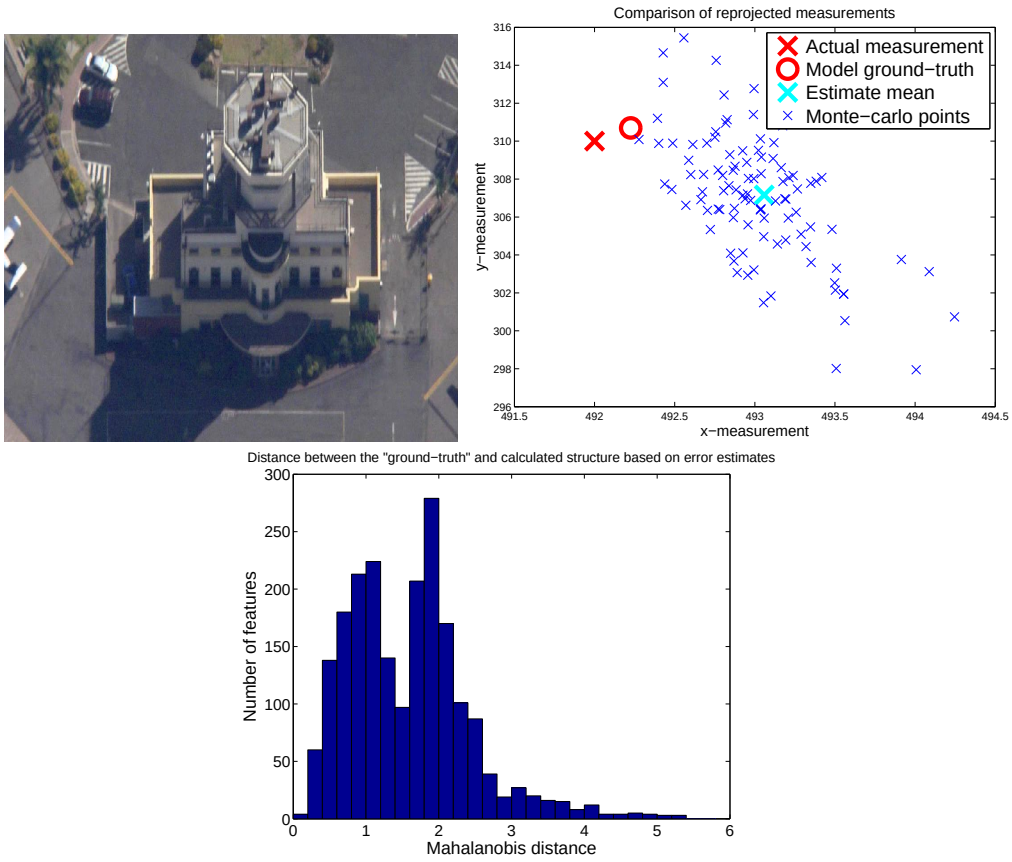

Figure 27: *a) Image from Parafield sequence, b) Error estimates for the reprojection of a point, c) Histogram of Mahalanobis distances of "ground-truth" from estimate.*

dinosaur data, substantially fewer points were lost due to obscuration. 51 of the features were present in all 60 frames. As a result, the missing data is less of a problem. Missing data is handled using the robust version of Tomasi-Kanade followed by 100 iterations of Shum's method.

When calculating the errors, ground-truth is not currently available for comparison. Therefore, the original measurements were used to estimate camera poses, structure and translations R, S and T which were rotated and scaled, and then treated as though they were the ground-truth. A new measurement matrix M' was then produced by randomly removing either the first or last half of each of the feature tracks, provided that the resulting track appeared in at least 4 of the frames. The estimation and Monte-Carlo simulations described above were then applied to generate the mean estimate R_o, S_o and T_o, as well as a set of noisy estimates, which were all aligned and registered (on a point-by-point basis, rather than as an image) together. It was expected that the "ground-truth" should lie within the error ellipses around the mean estimate.

For one of the points tracked over the entire sequence, Figure 27b shows the observed measurement, the reprojected approximation using all of the known points, the average estimate based on half of the points, and the cloud of erroneous measurements. As the

actual measurement lies almost within the cloud of estimates, in this case the error estimate would be quite accurate. If the error were a Gaussian with parameters estimated from the Monte-Carlo runs, it would be expected that the average Mahalanobis distance between the "ground-truth" and the estimates would be 1.18. For this data, the median was measured to be 1.82 over the entire set of measurements, which means that there is an under-estimate in the error, but it is close enough to be useful. Figure 27c shows a histogram of the Mahalanobis distance of the 3D point measurements with respect to the Monte-Carlo simulated estimates. As this is 3D instead of 2D, the expected median Mahalanobis distance is now 1.53. The observed value is 1.58, indicating that the error estimate in the structure is better than for the model reprojection.

7 Conclusions

This report has described a number of techniques which have applications to the extraction of a 3D model from a video sequence taken by an airborne sensor. In Section 2, detectors of feature points and lines were discussed. The FAST corner detector of Rosten and Drummond [23] was found to give very good detection performance, and its speed makes it a candidate to replace the Harris detector. Line detectors were also discussed, however as good performance was not achieved with the line based factorisation described in Subsection 3.4 these methods have not proved useful in 3D model reconstruction. Possible uses might be in a factorisation method incorporating both points and lines, but this requires further study.

Section 3 discussed structure from motion techniques, with an emphasis on methods for dealing with missing data. A number of methods were tested on the standard dinosaur data set. Firstly, it was found that Kanade and Tomasi's hallucination method [31], which is often used as an initialising step for other algorithms, could be improved significantly by using a robust least squares fit. This could then be used to initialise Shum's method [30], which was the best of the tested methods from the literature.

Section 4 describes several methods for post-processing the 3D point positions produced by the factorisation step. This included a method for smoothing out errors in the point positions, and methods for segmenting the detected points into planar regions. None of these worked particularly well, and were superseded by the dense matching methods described in Section 5. Stereo dense matching methods use two images, and information about the camera geometry to estimate the depth of each pixel within the image. This largely ignores the detected corner points, and only uses the camera information to determine pairs of epipolar lines. However, the resulting 3D model using the suggested graph-cuts method [14] appears sufficiently good for the purposes of video registration.

Finally, Section 6 introduced a framework in which errors in the construction of the model, and any subsequent registration, can be evaluated. Due to the lack of ground-truth in the image sequences used, it was not possible to produce an exact error measure with which to compare the resulting error estimate. However, using simulated ground-truth based on the measurements indicated that the error estimate for the 3D positions was fairly reasonable.

References

1. A.Bobick and S.Intille, "Large occlusion stereo," International Journal of Computer Vision, Vol.33, No.3, pp.181–200, 1999.

2. Y.Boykov, O.Veksler and R.Zabih, "Fast approximate energy minimization via graph cuts," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.23, No.11, pp.1222–1239, 2001.

3. Y.Boykov and V.Kolmogorov, "An experimental comparison of min-cut/ max-flow algorithms for energy minimization in vision," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.26, No.9, September 2004.

4. J.Burns, A.Hanson and E.Riseman, "Extracting straight lines," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.8, No.4, July 1986.

5. P.Chen and D.Suter, "Recovering the missing components in a large noisy low-rank matrix: Application to SFM". IEEE Transactions on Pattern Analysis and Machine Intelligence", Vol.26, No.8, pp.1051–1063, 2004.

6. T.Cooke, "A Radon transform derivative method for faint trail detection in SAR imagery," DICTA99 conference proceedings, pp.31–34, December 1999.

7. T.Cooke and R.Whatmough, "Detection and tracking of corner points for structure from motion," DSTO Technical Report, DSTO-TR–1759, December 2005.

8. T.Cooke, R.Whatmough, N.Redding and E.El-Mahassni, "An overview of geolocation of airborne video using 3D models," DSTO Technical Report, DSTO-TR-2001, January 2008.

9. E.El-Mahassni and T.Cooke, "A survey on the suitability of some recent 3D surface reconstruction algorithms for airborne sensor imagery," DSTO-TR–2064, October 2007.

10. L.Falkenhagen, "Depth estimation from stereoscopic image pairs assuming piecewise continuous surfaces," from *Image Processing for Broadcast and Video Production*, Paker and Wilbur (eds.), Springer series of workshops in computing, Hamburg, pp.115–127, 1994.

11. A.Fitzgibbon, G.Cross and A.Zisserman, "Automatic 3D model reconstruction for turn-table sequences, 3D structure from multiple images of large scale environments," in LNCS 1506, pp.155-170, 1998.

12. L.Ford and D.Fulkerson, "Maximal flow through a network," Canadian Journal of Mathematics, Vol.8, pp.399–404, 1956.

13. A.Goldberg and R.Tarjan, "A new approach to the maximum-flow problem," Journal of the Association for Computing Machinery, Vol.35, No.4, pp.921–940, October 1988.

14. A.Hornung and L.Kobbelt, "Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding," IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Vol.1, pp.503–510, 2006.

15. D.Huynh, R.Hartley and A.Heyden, "Outlier correction in image sequences for the affine camera," Proceedings of the 9th IEEE Conference on Computer Vision, Vol.1, pp.585–590, October 2003.

16. D.Jacobs, "Linear fitting with missing data for structure-from-motion," Computer Vision and Image Understanding, Vol.82, No.1, pp.57–81, 2001.

17. G.Koepfler, C.Lopez and J.Morel, "A multiscale algorithm for image segmentation by variational method," SIAM Journal on Numerical Analysis, Vol.31, No.1, pp.282–299, February 1994.

18. P.Kovesi, "Image features from phase congruency," Vidire: A Journal of Computer Vision Research, MIT Press, Vol.1, No.1, Summer 1999.

19. P.Kovesi, "Phase congruency detects corners and edges," Digital Image Computing: Techniques and Applications, DICTA 2003, Sydney, pp.309–318, 2003.

20. J.Li and N.Allinson, "A comprehensive review of current local features for computer vision," University of Sheffield (Unpublished), 2007.

21. C.J.Poelman and T.Kanade, "A paraperspective method for shape and motion recovery," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.19, No.3, pp.206–218, 1997.

22. L.Quan and T.Kanade, "Affine structure from line correspondences with uncalibrated affine cameras," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.19, No.8, August 1997.

23. E.Rosten and T.Drummond, "Fusing points and lines for high performance tracking," 10th IEEE Conference on Computer Vision, Vol.2, pp.1508–1515, October 2005.

24. E.Rosten and T.Drummond, "Machine learning for high-speed corner detection," European Conference on Computer Vision, May 2006.

25. E.Rosten, R.Porter and T.Drummond, "Faster and better: A machine learning approach to corner detection," personal communication, June 2007.

26. S.Roy and I.Cox, "A maximum-flow formulation of the N-camera stereo correspondence problem," Sixth International Conference on Computer Vision, Bombay, pp.492–499, 1998.

27. D.Scharstein and R.Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," International Journal of Computer Vision, Vol.47, pp.7–42, 2002.

28. D.Scharstein and R.Szeliski, "Middlebury college stereo vision research page."
`http://cat.middlebury.edu/stereo`

29. J.Shi and C.Tomasi, "Good features to track," Proceedings of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR'94), Seattle, June 1994.

30. H.Shum, K.Ikeuchi and R.Reddy, "Principal Component Analysis with missing data and its application to polyhedral object modeling", IEEE Transactions in Pattern Analysis and Machine Intelligence, Vol.17, No.9, pp.854–867, 1995.

31. C.Tomasi and T.Kanade, "Shape and motion without depth", Proceedings 3rd International Conference on Computer Vision, IEEE, pp.91–95, 1990.

32. C.Tomasi and T.Kanade, "Shape and motion from image streams: A factorization method – Part 3: Detection and tracking of point features," Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

33. C.Tomasi and T.Kanade, "Shape and motion from image streams: A factorization method – Full report on the orthographic case," Carnegie Mellon University Technical Report CMU-CS-92-104, March 1992.

34. C.Tomasi and T.Kanade, "Shape and motion from image streams under orthography : A factorization method," International Journal of Computer Vision, Vol.9, No.2, pp.137–154, 1992.

35. O.Troyanskaya, M.Cantor, G.Sherlock, P.Brown, T.Hastie, R.Tibshirani, D.Botstein and R.Altman, "Missing value estimation methods for DNA microarrays," Bioinformatics, Vol.17, No.6, pp.520–525, 2001.

36. O.Veksler, "Reducing search space for stereo correspondence with graph cuts," Proceedings of the British Machine Vision Conference, 2006.

37. S.Venkatesh and R.Owens, "An energy feature detection scheme," International Conference on Image Processing, Singapore, pp.553–557, 1989.

38. R.Whatmough, "Extracting the shape of a target from an image sequence with incomplete metadata," DSTO-TR-xxxx , 2007.

39. R.Whatmough, "Error analysis of shape from motion extraction with incomplete metadata," DSTO-TR-xxxx , 2007.

40. C.Zitnick and T.Kanade, "A cooperative algorithm for stereo matching and occlusion detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.22, No.7, pp.675–684, July 2000.